BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF BUSINESS AND MANAGEMENT
FAKULTA PODNIKATELSKÁ

INSTITUTE OF INFORMATICS
ÚSTAV INFORMATIKY

DESIGN AND IMPLEMENTATION OF A DIGITAL LICENCE MANAGEMENT SYSTEM
NÁVRH A IMPLEMENTACE SYSTÉMU PRO SPRÁVU DIGITÁLNÍCH LICENCÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — Adam Baliak
AUTOR PRÁCE

SUPERVISOR — Ing. Jan Luhan, Ph.D., MSc
VEDOUCÍ PRÁCE

BRNO 2019

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF BUSINESS AND MANAGEMENT

Bachelor's Thesis Assignment

Institut:	Institute of Informatics
Student:	**Adam Baliak**
Degree programm:	System Engineering and Informatics
Branch:	Managerial Informatics
Supervisor:	**Ing. Jan Luhan, Ph.D., MSc**
Academic year:	2018/19

Pursuant to Act no. 111/1998 Coll. concerning universities as amended and pursuant to the BUT Study Rules, by the Director of the Institute, you have been assigned a Bachelor's Thesis entitled:

Design and Implementation of a Digital Licence Management System

Characteristics of the project issues:

Introduction
Aim of the Thesis
Theoretical Background
Problem Analysis and Current Situation
Proposals and Contribution of Suggested Solutions
Conclusions
References
Appendices

The objectives to be achieved:

Design and implement an information system for effective management of licences for digital products, which are provided by the company. The information system should be based on company requirements. Moreover, the system will be responsible for authorising the update requests, version management and to provide usage statistics of individual product versions.

Literature on the topic:

HUNTER II, T. Advanced Microservices: A Hands-On Approach to Microservice Infrastructure and Tooling. Berkeley: Apress, 2017. 181 p. ISBN 978-1-4842-2886-9.

PAPER, D. Web Programming for Business: PHP Object-Oriented Programming with Oracle. 1st ed. New York: Routledge, 2015. 280 p. ISBN 978-0-4158-1804-9.

SCHWARTZ, B. and col. High performance MySQL: Optimization, Backups, Replication, and More. 2nd ed. Sebastopol: O´Reilly Media, 2008. 684 s. ISBN 978-0-596-10171-8.

SKLAR, D. a J. POKORNÝ. PHP 7 - Praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. 1. vyd. Brno: Zoner press, 2018. 368 s. ISBN 978-80-7413-363-3.

VAN TILBORG, H. C. A. Encyclopedia of Cryptography and Security. 1st ed. New York: Springer, 2005. 684 p. ISBN 978-0387-23473-1.

The deadline for submission for the Bachelor's Thesis is given by the Schedule of the Academic year 2018/19.

In Brno, 28. 2. 2019

L. S.

doc. RNDr. Bedřich Půža, CSc. Director of the Institute	doc. Ing. et Ing. Stanislav Škapa, Ph.D. Dean

ABSTRACT

This bachelor thesis deals with the analysis, design, and implementation of an information system for a company specialising in software development. The purpose of the information system is to provide a feasible platform for management and provide usage statistics of various digital products, which require licence key to be activated and updated. It will be able to generate licence keys and authorise update requests originating from customer systems.

ABSTRAKT

Táto bakalárska práca sa zaoberá analýzou, návrhom a implementáciou informačného systému pre spoločnosť zaoberajúcu sa vývojom softvéru. Zmyslom tohto informačného systému je slúžiť ako platforma pre manažment a poskytovať štatistiky využívania rôznych digitálnych produktov, ktoré vyžadujú licenčný kľúč na to aby boli aktivované a dostávali aktualizácie. Ďalej bude tento systém generovať licenčné kľúče a autorizovať aktualizačné požiadavky, prichádzajúce z klientskych informačných systémov.

KEYWORDS

information system, web application, licence management, PHP, SQL, database

KĽÚČOVÉ SLOVÁ

informačný systém, webová aplikácia, správa licencií, PHP, SQL, databáza

ROZŠÍRENÝ ABSTRAKT

Cieľom práce bolo navrhnúť a implementovať informačný systém, ktorý by firme umožnil efektívne spravovať a vydávať licencie k digitálnym produktom, ktoré firma vyvíja. V tejto dobe sa jedná o programy, ktoré rozširujú funkcionalitu internetových obchodov a sú populárne medzi malými obchodníkmi. Nápad realizovať takýto systém, vznikol pri práci na internetových obchodoch pre túto skupinu klientov. Stále sme sa stretávali s rovnakými požiadavkami na rozšírenie funkcionality – najžiadanejšie boli integrácie platobných brán rôznych bánk. Preto sme sa zamerali na vývoj týchto riešení, ale stále sme potrebovali nástroj, ako predané produkty spravovať a zamedziť ich šíreniu. Taktiež, vydávanie aktualizácií manuálne a písanie jednotlivým zákazníkom trvalo nejakú dobu. To bol hlavný dôvod, prečo som do systému zahrnul aktualizačný modul a produkty u zákazníkov boli upravené tak, aby sa aktualizovali automaticky cez tento licenčný systém.

Táto práca je písaná v anglickom jazyku, nakoľko firma má aj zahraničných zákazníkov a partnerov, ktorých môže implementácia tohto riešenia zaujímať.

Popis riešenia

Pre firmu som navrhol kompletný systém na spravovanie licencií pre digitálny obsah. V súlade s požiadavkami je tento systém zameraný na jednoduchosť používania a na bezpečnosť. Pre uľahčenie používania som použil CSS knižnicu Bootstrap 4. Vďaka použitiu tejto knižnice je tento systém responzívny a teda sa dokáže prispôsobiť zariadeniu, na ktorom sa zobrazuje. Pohodlie užívateľa zvyšuje využívanie asynchrónnej komunikácie na jednotlivých stránkach systému, čo rapídne znížilo potrebný počet obnovení stránok pri práci so systémom, a teda aj urýchlilo pracovné postupy. Pri návrhu systému bol kladený dôraz na bezpečnosť systému – ošetrili sa najrozšírenejšie zraniteľnosti systémov podľa konzorcia OWASP. Pri vytváraní nových používateľov v systéme je potrebné pri každom používateľovi vyplniť jeho telefónne číslo, na ktoré bude dostávať jednorazové prihlasovacie kódy pri pokuse o vstup do systému. Toto opatrenie bolo zavedené kvôli potrebe zamedziť neautorizovanému prístupu do systému, ktorý je definovaný v zmluve na základe, ktorej sa prístup do systému vytvára, a taktiež kvôli tendencii používateľov používať slabé prístupové heslá. Všetky vstupy od používateľov sú kontrolované najprv u užívateľa pomocou JavaScriptu a po úspešnom odoslaní nasleduje druhá kontrola na

servery, kde sa zo vstupu odstraňujú aj prípadné zakázané znaky tak, aby nedošlo k neoprávnenej manipulácií s databázou a bola zaručená ako integrita, tak aj dôvernosť dát.

Licencie, ktoré firma vystavuje sú vždy viazané na konkrétny produkt a konkrétnu doménu, na ktorej je spustený klientský systém, do ktorého sa produkt inštaluje. Túto doménu je možné neskôr zmeniť a to odoslaním požiadavky na podporu obchodníka, ktorý daný produkt zákazníkovi predal a ktorý následne prepošle túto požiadavku do firmy ktorá prevádzkuje tento systém. Toto riešenie bolo navrhnuté s cieľom maximalizovať zisk z produktov pre firmu. V praxi som sa často stretával s tým, že jeden zákazník mal viacero systémov (internetových obchodov), ale za podobné produkty nechcel platiť viac ako jeden krát, ak nebola obmedzená funkcionalita daného produktu.

Najväčšou výzvou bolo navrhnutie procesu validácie aktualizačných požiadaviek. Bolo potrebné navrhnúť také riešenie aby systém vedel spoľahlivo autentifikovať klientsky systém, z ktorého požiadavka prišla a následne túto požiadavku autorizovať. Predtým ako klientský systém pošle aktualizačnú požiadavku je potrebné vyžiadať si podpis systému, ktorý je závislý na konkrétnom produkte a doméne z ktorej požiadavka prichádza. IP adresa tejto domény, získaná z A (AAAA) DNS záznamu je porovnaná s IP adresou z ktorej prichádza požiadavka o podpis. Ak sa tieto IP adresy zhodujú tak požiadavku považujeme za autentickú a systém vydá podpis systému, ktorý sa použije pri aktualizačných požiadavkách. Tento podpis slúži ako digitálny podpis a ide o HMAC hash v ktorom sú overované údaje a kód produktu. Tento proces je popísaný na Obrázku 3.7.

V systéme je integrovaný aktualizačný server, ktorý bol upravený aby autorizoval požiadavky cez licenčný systém. Licenčný systém umožňuje zobrazenie štatistík jednotlivých produktov a ich vizualizáciu vo forme prehľadných grafov. Stránka systému zobrazujúca štatistiky produktu je viditeľná na Obrázku 3.9.

Náhľad systému je k dispozícií v prílohách tejto práce.

Zrhnutie dosiahnutých výsledkov

Návrh systému trval približne 15 hodín a implementácia 90 hodín. Celkový čas strávený návrhom a implementáciou je teda 105 hodín. Systém bol vyvinutý interne pre firmu, takže celkové náklady na systém teda predstavujú náklady obetovaných príležitostí, keďže som sa mohol v tom čase venovať práci na inom projekte. Pri priemernej cene 200Kč za hodinu mojej práce, sú tieto náklady vo výške 21 000Kč. Systém má malé

požiadavky na hardvér a nevyžaduje ďalšiu investíciu do infraštruktúry firmy. Všetok sofvér, ktorý je v systéme použítý, má otvorený zdrojový kód, čo taktiež pomohlo znížiť náklady na vývoj systému. Firma sa ďalej rozhodla prevádzkovať systém na subdoméne, ktorú už firma vlastní, takže ani tu už nevznikli ďalšie náklady.

Na záver považujem za dôležité spomenúť fakt, že žiadny softvér, ktorého zdrojový kód sa nachádza u klienta, nie je možné ochrániť tak, aby nebolo možné upraviť jeho funkcionalitu, ktorá overuje oprávnenia k používaniu. Z týchto dôvodov bol systém overovania a aktualizovania navrhnutý tak, aby zamedzil používaniu na viacerých doménach. Pre klienta je hlavnou motiváciou ku kúpe produktu pridaná hodnota automatických aktualizácií, ktoré systém poskytne len oprávneným požiadavkám.

BIBLIOGRAPHIC CITATION

BALIAK, Adam. *Design and Implementation of a Digital Licence Management System* [online]. Brno, 2019 [cit. 2019-05-12]. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/119572. Bachelor's thesis. Vysoké učení technické v Brně, Fakulta podnikatelská, Institute of Informatics. Supervisor Jan Luhan.

DECLARATION OF ORIGINALITY

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Mr. Ing. Jan Luhan Ph.D., MSc. I also declare that I did not breach any copyright in sense of Act. No. 121/200 Coll. on Copyright Law and Rights Related to Copyright and on amendment of Certain Legislative Acts. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

. . . . . .

Adam Baliak

Brno, May 12, 2019

ACKNOWLEDGEMENTS

I would like to express gratitude to my supervisor Mr Ing. Jan Luhan Ph.D., MSc., for his guidance and advice while I was creating this thesis. I am also grateful to my family and friends who supported me along the way.

CONTENTS

INTRODUCTION

Information systems are now powering most of the businesses around the world. With the number of e-commerce sites rapidly increasing in recent years, there is demand after solutions which can power these businesses. Providing quality support to the customers and offering them solutions should be a number priority for companies specialising in information technologies.

Designing a solution which will create value for the client is not an easy task. One needs to understand the underlying motivation and needs of the client. Since I have been working on the development of information systems for the past four years, I wanted to prove to myself that I can design a working solution on my own. I also wanted to see my solution be used in a production environment and live its own life as long as it can bring value to the company.

I combined the knowledge which I have acquired while studying system engineering at the Faculty of Bussiness and Management with the practical knowledge I have gained while developing solutions for clients. Most of these clients wanted small e-commerce sites, occasionally there was a request for a more complex information system. After seeing that clients are dealing with the same problems every time, I thought, why not provide them with the solutions they need? And so I started working on a platform which could handle all of the logistics of delivering small pieces of software - known as plugins - to the clients. From this idea, a Licence Management System concept was created.

The goal was to create a system which the company could use to issue licences to its clients, push updates and easily add new products as they are being developed. I can see that in the near future the system will be extended with integration of external e-commerce site which will resell offered products. The idea behind the system is to decrease the costs associated with marketing and instead focus on the development of the new products since the company has limited resources available.

I tried to approach the problem presented in this thesis a little bit differently. I knew exactly what the problem was so I researched the best practices and how to approach it in a way that it will be future-proof. Those principles and methods are described down to the level which I thought is necessary to understand this problem. The knowledge is spread

across a wide spectrum of technologies and concepts. I described only that knowledge which directly impacted the development of the system.

This bachelor's thesis is divided into four chapters. The first one describes the theoretical background and other knowledge relevant to the topic of the thesis. Chapter 2 analyses the current state in the company and analyses requirements put on the information system. Chapter 3 describes how the system was designed and subsequent implementation of the designed solution. The system is also evaluated in this chapter from different perspectives and plans are laid out for the future development of the system.

AIM OF THE THESIS

The primary objective of this bachelor thesis is to analyse, design and implement an information system for a company specialising in software development. The information system will be used as a platform for managing licences for digital products of the company. Emphasis should be given especially on application security as the system will contain confidential data. The company has provided us with detailed requirements, these will all be addressed after a research of the current state is made and informed decisions could be made.

Subsidiary aims are:

1. analyse the current state in the field of databases and web applications development, with emphasis on security
2. analyse requirements of the company
3. design a solution which satisfies all of the requirements
4. implement the designed solution
5. evaluate the benefits of the new system to the company

Methodology

I will use methods and technical knowledge which I have gained during my studies in the Czech Republic and while I worked on various e-shops, information systems and web presentations as a contractor. I also used the knowledge which I have gained while working as a backend developer for company inQool a.s. These methods are described in relevant parts of this thesis.

1 THEORETICAL BACKGROUND

In this chapter, I am describing knowledge which I find essential for understanding this thesis. However, it is by no means an extensive extract. My goal was to include just the essential information, therefore only knowledge which I considered advanced, or was new to me, will be described in depth here. Otherwise, description and a reference will be provided.

1.1 Data and Information

Data has no meaning or value without a context. Data can be viewed as potential information because if data is used in decision making it becomes information. However, there is no linear relationship between data and information, since data can also emerge from the information, and information from knowledge. [1]

Information is data that have been shaped into a form that is useful in decision making or has any other value to the person. In the most general meaning information can be perceived as a description of something - may it be some system or a real-world environment. Information reduces the ambiguity of the system. [1]

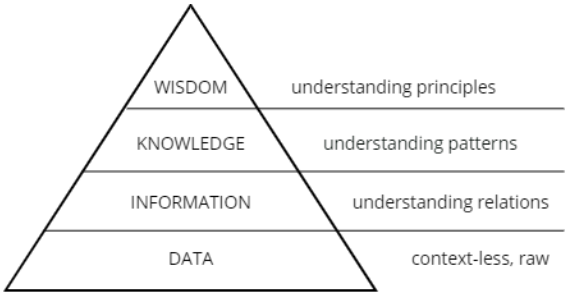


Figure 1.1: The information hierarchy (Source: based on [1])

The hierarchy is pretty simple. Data is everything that is around us - for example a web page, billboard, newspaper article. Information requires an ability to perceive and understand the data because that is what puts data into context. Knowledge is the ability to work with information and recognise patterns occurring in it. Wisdom is a combination of intuition and experience, in other words, it allows one to understand the underlying principles. [1]

1.2 Exchanging Information

With the growth of information systems and the increasing amount of data collected there was a need for an effective way of data exchange. Since the traditional way of data exchange - psychical deliveries - we unfeasible. Therefore, the internet and world wide web were born. These technological inventions enabled enormous growth of business around the world.

1.2.1 Internet

The internet is a constantly evolving network of interconnected devices. Conceptualized in 1969 by the Advanced Research Projects Agency, and originally used to exchange data between research laboratories. Later, in the early 1990s, the World Wide Web was created at CERN. Scientists were producing such a large amount of data that was unviable to distribute it across the world to all of the participating scientists. [23]

Computers are connected to the internet via their internet service providers. These providers are connected via massive backbone networks, which are also in place underwater, interconnecting world continents. Every computer connected to the internet is assigned an address, but not every address can be reached by anyone. There are public and private addresses. Public addresses are mostly assigned to servers because they need to access by anyone to provide services. [23]

1.2.2 Client-server Model

This model describes two roles that can be played by participating processes. Service consumer - client, and a service provider - server. The client is in RFC2616 defined as „A program that establishes connections for the purpose of sending requests.“. The server is defined as an „An application program that accepts connections in order to service requests by sending back responses.“ Any program can serve both as a client and a server. For example, if a server has to retrieve results from another service it then becomes also a client. [28]

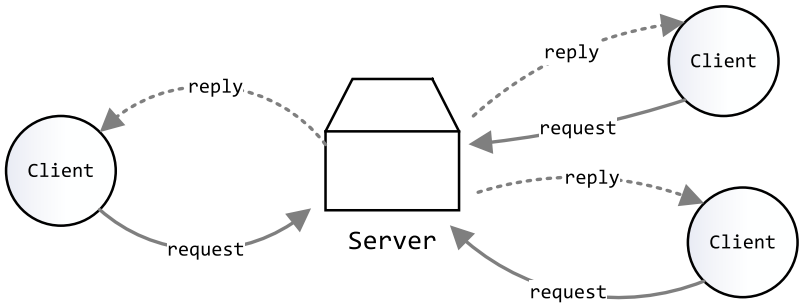


Figure 1.2: Client-server communication scheme (Source: based on [28])

1.2.3 Transport Protocols

When computers are interconnected they also need to know how to communicate with each other and exchange data. This is why transport protocols have been defined. One of the first protocols defined has been a Hypertext Transfer Protocol (HTTP). Nowadays, there are more protocols with each having its own niche use, but HTTP is the protocol which runs the whole World Wide Web. [23]

HTTP

Hypertext Transfer Protocol (HTTP) is a communication protocol which describes how a request and response has to look like. HTTP makes use of the client-server model (section 1.2.2). When a client request resources from the server, the server will respond with a status code and either return the requested resource or deny the request. A simplified diagram of HTTP request flow is shown in Figure 1.3. [23]

HTTPS

HTTP-secure is a secure version of the original HTTP. Originally it used Secure Socket Layer (SSL) atop of HTTP but in recent years, usage has shifted towards a more secure approach. Transport Layer Security (TLS) is the only HTTPS standard which is considered secure as SSL has discovered security vulnerabilities. TLS ensures connection privacy and message integrity by encrypting transported data. A detailed description of how TLS works is available as an RFC8446[1] document. [29]

[1]RFC8446: The Transport Layer Security (TLS) Protocol Version 1.3

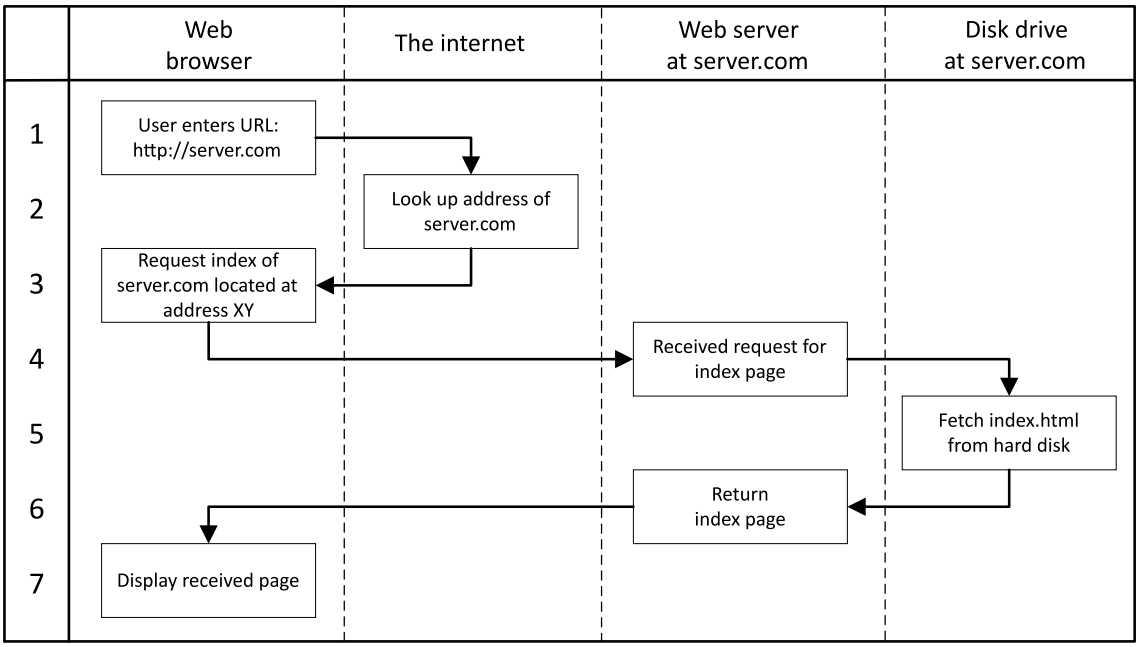


Figure 1.3: Simplified flow of a HTTP request (Source: based on [23])

1.2.4 JavaScript Object Notation (JSON)

JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format. JSON defines a small set of formatting rules for the portable representation of structured data. JSON can represent primitive types such as string, numbers and two structured types - objects and arrays. [4]

```
{
  "movie": {
    "title": "Fantastic movie II",
    "year": "2019",
    "rating": 3,
    "actors": [
      "James Arthur",
      "James Bond"
    ]
  }
}
```

Figure 1.4: JSON encoded data example (Source: original work)

JSON is language independent yet is supported in virtually every language. That is why is it widely used as a way to return easily parseable results from backend services or to communicate between services. [4]

1.2.5 Asynchronous Transfer

In the past, asynchronous transfer was referred to as AJAX. It stands for Asynchronous JavaScript and XML. It is a method of asynchronous transfer of data between browser and server in the background. In the background means that the browser does not have to wait to request to finish loading as data will be displayed dynamically. Typically, it communicates with exposed API endpoints on the server and exchanges data encoded in JSON. It eliminates the need for page refresh after every user action and makes even a simple page dynamic. Pages built with asynchronous actions are conceived by users more like self-contained applications with better user interface and responsiveness. An example of a webpage that depends heavily on asynchronous transfers is Google Maps[2] which would otherwise require a Flash embed for this user experience. [22]

1.3 System Model and Architecture

There are many ways in which a software application can be designed. This is especially true with web applications, where numerous application components are used and communicate together. The application model then becomes quickly bloated and we need to choose one architecture style and divide our application into several logic sections. One of the most common architecture styles is the n-Tier architecture which is described below, in section 1.3.1. [11]

When the system is (being) designed it is helpful to visualise how it interacts with other systems and its users. One of the simplest yet most descriptive is the C4 model, described in section 1.3.3. It provides insight into external interactions and also into internal structures of the application. [11]

1.3.1 n-Tier Architecture

This architecture model divides the system into multiple layers, typically three. The first one is a presentation layer or commonly referred to as frontend, which focuses mainly on presenting information and receiving input from users. This layer should not contain any business logic, because it could be changed by a knowledgeable user. Data, meanwhile, are stored on one or more data servers in the data layer. This layer contains only logic

[2]Google Maps - `https://maps.google.com`

which is necessary to access the data and maintain its integrity - it could be for example a set of predefined procedures, transactions, triggers. All of the business logic is held at the middle layer - the backend - which processes the user input, retrieves and returns requested data when the user accessing data is authorized to do so. [21]

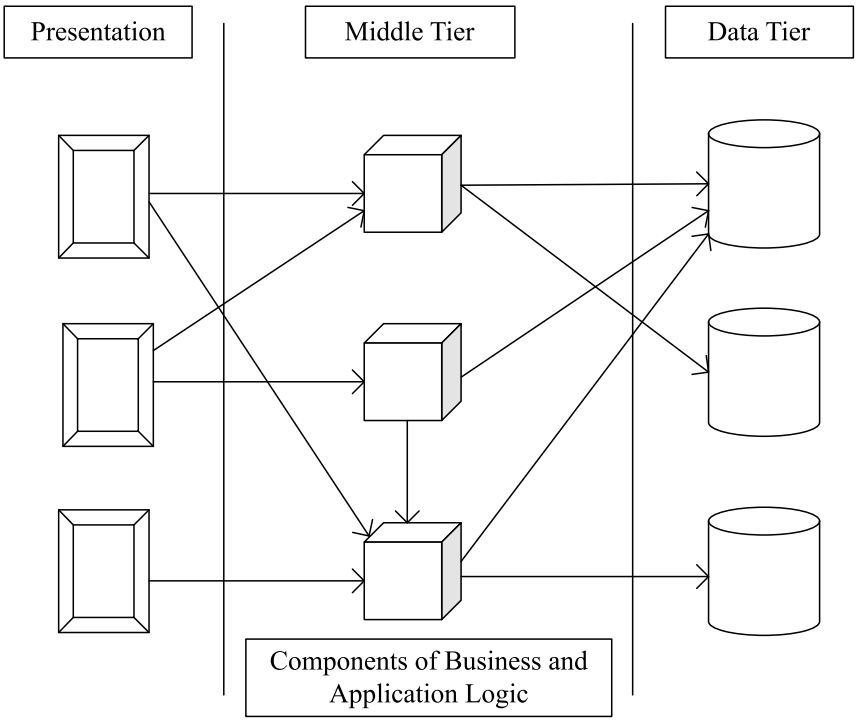


Figure 1.5: n-Tier architecture (Source: based on [21])

Splitting the application into different layers, and keeping the business logic only at one is beneficial for integrity and keeping the application code simple. The simplicity comes in a way that when a bug is found, a software developer only needs to look for it in one place instead of multiple systems. [21]

1.3.2 Entity-relationship Diagram

Probably the most common data model, it is an entity-relationship (ER) model which is created by combining several linear models. This connection is not permanent and is created only when we are accessing data from multiple connected tables. ER diagram not only shows data of the inspected object but also their relationships. The building block of this model is a relation (described in Section 1.6.1) which can be simply represented as a table. This table contains columns (attributes) and rows (instances). [15]

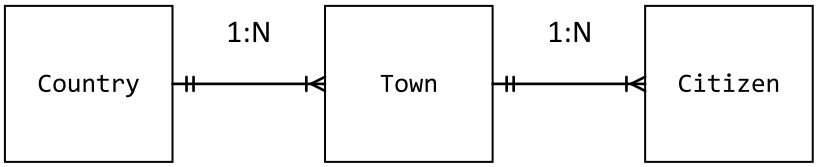


Figure 1.6: Example of an ER diagram (Source: original work)

There are many ways how the relationship can be shown in the ER diagram. In this thesis, I am using the Crow's foot method of notation.

1.3.3 C4 Model

The C4 model provides a way of simply describing the software system. Systems are often a complex structure of containers which communicate with each other. When you look at a system as a whole it is only one container which can communicate with other external systems. This view of the system is in the C4 model called the context view. A container view provides a more complex overview of the system. It breaks it down to individual containers within the system and shows how they interact with other containers. The C4 model can be then used to show individual components of each container - the container view. The last level of the C4 model is the class view which shows individual classes in a component in the container in the system. [5]

By using this model one can create a detailed model of the software system. Because a person which is new to the system can start at a general context view of the system and then dive into the specifics, this model makes it easy to understand how the system functions. In other words, the hierarchy of the C4 diagrams provides different levels of abstraction, each of which is relevant to a different audience. [11][5]

1.4 Frontend Technologies

This section describes technologies which are used in the presentation layer of applications. These technologies are client-side and they define how will the application look and behave from the user's point of view.

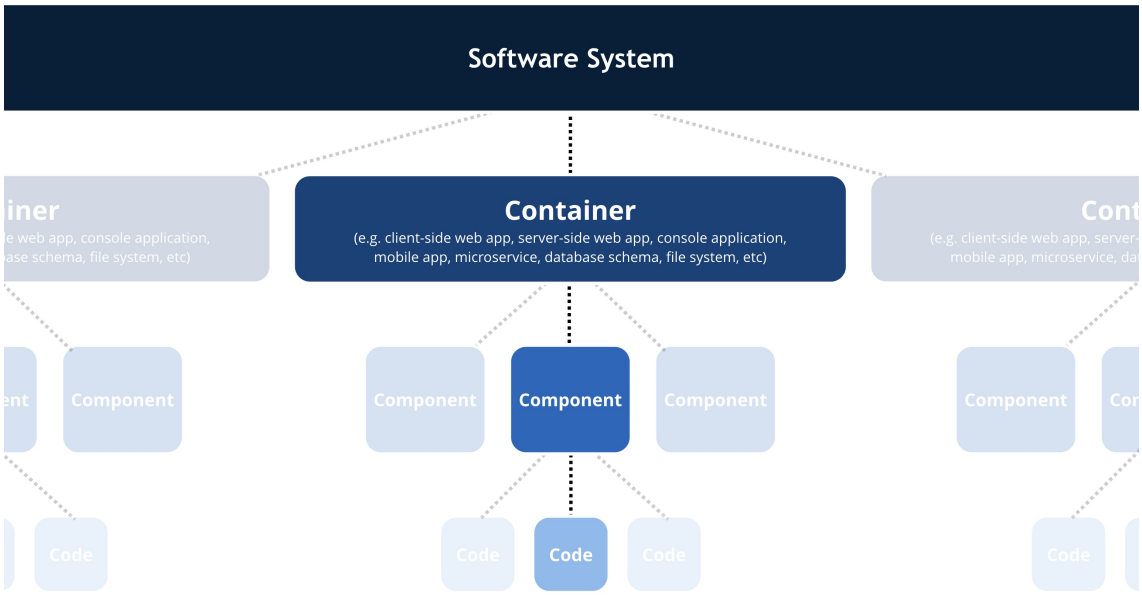


Figure 1.7: A software system structure (Source: [5])

1.4.1 Hypertext Markup Language (HTML)

„The Hypertext Markup Language (HTML) is a data language designed to present mostly static data inside a generic user interface application or browser.“ [8]

1.4.2 Cascading Style Sheets (CSS)

„Cascading Style Sheets are a technology that allow for the specification of how an HTML document will be displayed. It is a presentation specification mechanism not a programming language, and as such it augments HTML but does not fully compensate for the features HTML is lacking.“ [8]

CSS can be used to add style to the webpage simply by inserting required styling statements into the HTML <head> tag or effectively anywhere in the code (although this is considered bad practice since it makes code less readable). An example of such a statement can be found in Figure 1.8. CSS is in active development since its conception in 1996. Currently, a new major version - CSS3, in development since 2001 - is currently being widely adopted by browsers. CSS3 enables the use of advanced features such as new selectors, multiple backgrounds and, for the first time a functions which can be used to do simple style calculations. [23]

```
<style>
  body{
    background-color: blue;
    color: yellow;
  }
</style>
```

Figure 1.8: Example of CSS rule (Source: original work)

1.4.3 JavaScript

JavaScript is a powerful programming language that contains constructs which are missing in basic HTML. JavaScript creates an object model of the webpage and it is able to augment it and provide additional functionality. Therefore, JavaScript is commonly used to create „live“ pages which do not need to reload after every user action. [8]

Data Validation

Wherever there is a user input there is a need for data validation. This can be done at a receiving server (and should be done anyway) but one can save a one or two request by pre-validating data locally in the client's browser. Most common validations are a format of a phone number, postcode, date etc.. This validation ensures that data are in the expected format before they are sent to the server. [7]

1.4.4 jQuery

jQuery is an open-source library extending the JavaScript programming language. It provides an abstraction layer over the object model and enables quicker and easier object manipulation. On top of ease of manipulation, it provides a framework for some of the great JavaScript constructs like AJAX (see section **??**) and normalizes the behaviour of various browsers, so the user experience is consistent. [7]

1.5 Backend Technologies

Backend technologies are responsible for everything that the user does not get to see. In the n-Tier architecture, this the middle tier and it contains business logic and is responsible for processing user's request and returning valid responses. This section does not deal with the hardware aspect of the backend since it is out of the scope of this thesis.

1.5.1 Hypertext Preprocessor (PHP)

PHP is a scripting language that extends the functionality available in HTML. It is commonly used in dynamic pages whose content differs based on selected criteria - for example, if the current user is logged in, what permissions that user has and more. PHP is most often included in HTML and while this is technique is simple and fast it quickly becomes unsustainable to maintain the code. This problem can be solved by separating business logic from presentation templates. [23]

When a request for a page is received on a server, the web server checks if the source code of that page contains PHP code. If it does, the page is parsed by PHP processor running as a web server module or as a standalone processor such a PHP-FPM. The processor then takes care of computing what to display, makes queries on the database - if they are needed - and then it renders and sends the page back to requesting user. Simplified schema of how the requests are processed is shown in Figure 1.9. [23]

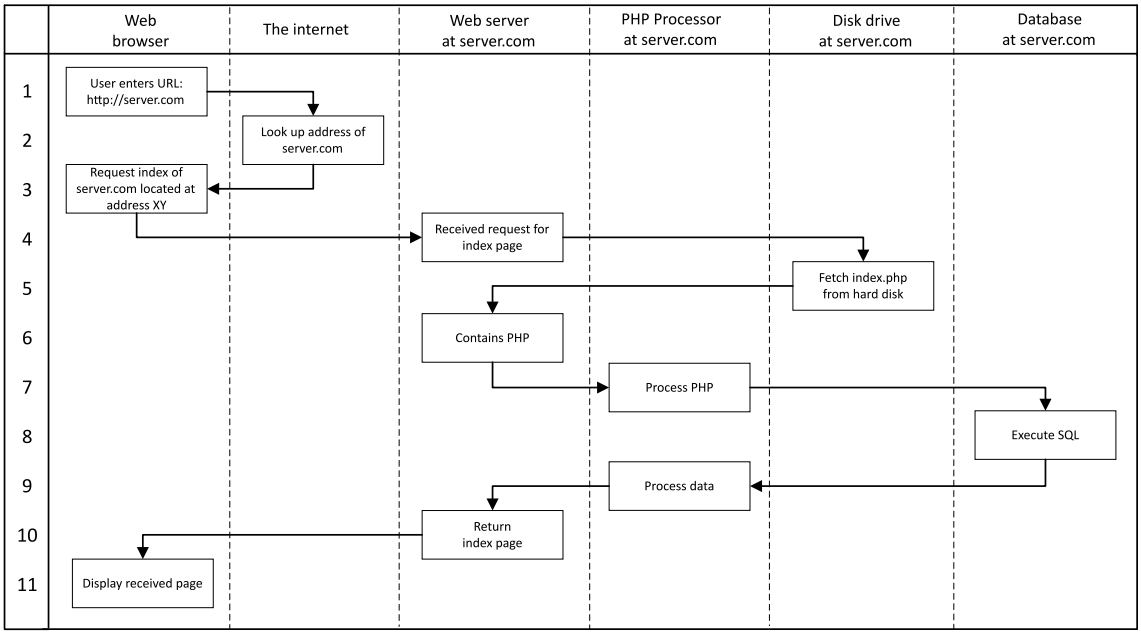


Figure 1.9: Flow of PHP request and response (Source: based on [23])

1.5.2 Universally Unique Identifier (UUID)

A UUID is 128 bits long, and can guarantee uniqueness across space and time. The main benefit of using UUIDs is that no global authority is required to manage them and yet they can guarantee uniqueness. What is more, they support high allocation rates - up to

10 million per second, and with a fixed size of 128 bits they are also reasonably small. UUIDs are commonly used as primary keys in databases. [19]

The complete definition of UUID is available as RFC4122 ([19]).

1.6 Database

We can imagine a database as a collection of data which describe the real world. Various database engines then provide different ways of data manipulation and searching. Databases are divided into various categories by intended usage. Relational databases are the most common and they can be used to power business on a day-to-day basis. A more specialized category are analytical databases. They are used to retroactively analyse collected data and provide insights about them to the managers. These databases are optimized for queries on big datasets. [13]

1.6.1 Data Model

Data model defines objects and their attributes, relationships in relational database. Having a proper data model is paramount to the function of the database. [23]

This section describes main objects and processes in the data model.

Entity and Attributes

An entity is something about what we want to store information. For example, it can represent a psychical object such as a car, or it can be an abstract object such as a medical record. Entities are specified by attributes, and every entity has its attributes. For instance, a car (entity) can have many characteristics (attributes) such as colour or VIN[3] code. [10][6]

Relation

Two-dimensional table with no repeating groups is called a relation. That means if we intersect row and column, we get only one value. The terms relation and table are often used interchangeably. In a well-designed database, each table represents one entity. [10]

[3]VIN – Vehicle Identification Number

Relationships

Relationships define how entities relate to each other. For example, a customer can have many orders, but each order can be assigned only to one customer. In this case, both customer and order are entities with a relationship between them. There are three types of relationships as described in Table 1.1. [10]

Relationship		First entity	Second entity
One-to-one	1:1	1 instance	1 instance
One-to-many	1:N	1 instance	1 or more instances
Many-to-many	N:M	1 or more instances	1 or more instances

Table 1.1: Entity relationships overview (Source: based on [10])

Decomposition

Decomposition is the process of breaking down complex relationships. Since the Many-to-Many relationship can not be represented (stored) in the table correctly, we need to create new table instead. [23]

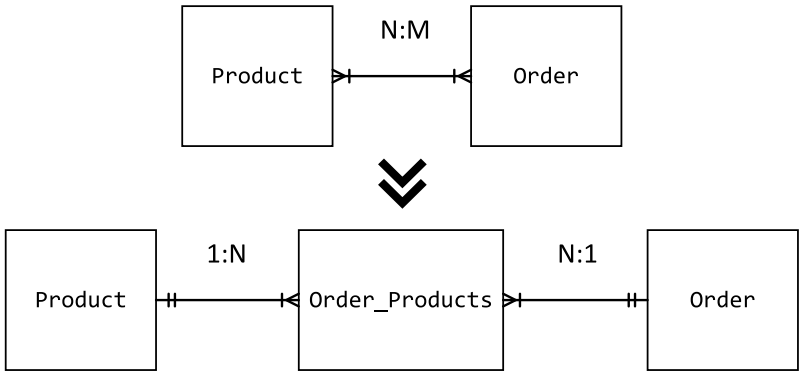


Figure 1.10: Example of an decomposition (Source: original work)

For example look at Figure 1.10, there are two entities, products and orders. One product can be in multiple orders and orders can have a multiple products. We need therefore to know which products are in which orders. [23]

1.6.2 Data Normalization

Normalization is the process of separating data into tables and creating primary keys. The main goal of normalization is that there are no duplicate data stored in the database

since storing duplicate data is inefficient. The main risk of storing such data is that there is a possibility of not updating it everywhere and thus creating ambiguity. E. F. Codd, who is considered the inventor of the relational model created three basic schemas called First, Second and Third Normal Form. By conforming to these schemas we will have an optimally balanced database. [23]

First Normal Form

The First Normal Form takes care of duplicate data across multiple columns. The database must conform to these requirements to satisfy the First Normal Form. There should not be repeating columns containing the same kind of data, every column should contain a single value and every row should have a primary key to uniquely identify itself. [23]

Second Normal Form

The Second Normal Form deals with redundancy across multiple rows. The first requirement of the Second Normal Form is that the database has to already be in a First Normal Form. However, to conform with the Second Normal Form there should not be columns with data that are repeated in different places, and if they are they need to be moved into their own tables. [23]

Third Normal Form

To achieve the Third Normal Form, the database has to be in the Second Normal Form and that data that is not directly dependent on the primary key, but it depends on another value in the table should be, according to dependence, also moved to a separate table. The Third Normal Form is considered to be a strict one and usually it is enough to conform to the first two normal forms. However, conforming with as many forms as possible is beneficial for the future as it helps with maintaining the database in the long run. [23]

1.6.3 Structured Query Language (SQL)

Structured Query Language (SQL) is the language used to make queries on the database. It consists of multiple parts and includes management tools, object creation and searching or updating of data. It is a declarative programming language which means that SQL

has to be included in another one, procedural language. Generally, queries can only be performed on a database after a successful authorisation. [17]

1.7 Application Security

Security is the protection afforded to the system to ensure that it maintains the three core attributes - Confidentiality, Integrity and Availability - known as a CIA triad. These three concepts are fundamental security objectives for both data and information services. Some fields of information security, however, recognize two more security attributes: Authenticity and Accountability. [30]

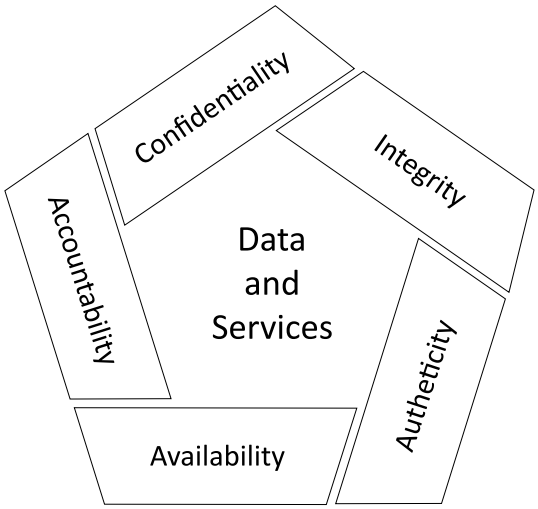


Figure 1.11: Essential security objectives (Source: based on [30])

Confidentiality

Data confidentiality assures that confidential information is not made available or disclosed to unauthorized individuals. Privacy assures that individual has control over what related information is collected and stored, to whom and how it will be disclosed. Loss of confidentiality is the unauthorized disclosure of information. [30]

Integrity

Data integrity assures that data - both at rest and in motion - are changed only in an authorized and specified manner. System integrity assures that the system performs its intended function without unauthorized manipulation. Loss of integrity is the unauthorized destruction or manipulation of information. [30]

Availability

Assures that service is not denied to authorized users in a timely and reliable manner. Loss of availability is the disruption of access to the system, or information. [30]

Authenticity

Is the property of being genuine, verifiable and trusted. When the message or transmission is authentic we can be confident that users are who they say are and that the input arrived from a trusted origin. [30]

Accountability

Accountability requires the ability to trace initiator of any action on the system to a unique entity. Systems are therefore required to keep logs of events happening in the system in order to trace security breaches or to help in transaction disputes. [30]

1.7.1 Data Protection

In relation to application security, we recognize two types of data: data at rest and data in motion. Data protection is important, especially since the new data protection regulations are in place. There is a paradigm about what protection is enough, and although one can have a set of strict rules in place it is better to minimize effects of security breaches by limiting user permissions and storing data in a secure fashion. [20]

Data at Rest

Inactive data that is being stored on the server, such as databases used to store information about users, their password etc. This type of data can be protected by having security processes in place, such as encryption of the database. When the data is encrypted, keys used for that encryption should be stored separately from data and also updated on a

regular basis to minimize the risks. The best concept one can adopt is minimising the amount of sensitive data that is collected about a user. This in contrast to the increasing trend of collecting as much data as is possible about the user, but it is worth it because it minimises the impact of a security breach. [20]

Data in Motion

Data at Motion or data that is in transit are sent back and forth between user's browser, application or database. This type of data is easily protected but even more easily misused if not protected. Example of data at motion is when a user wants to sign in to a service. The user enters his password into the webpage which sends it over the internet afterwards. We have to think about protected before the user submits this form because while the data is in transit we can not do anything more to protect it. The simplest measure is to adopt HTTPS (described in Section 1.2.3) as this will protect data from third parties. This, however, is not a silver bullet solution as there are types of attacks that can circumvent this - for example a man in the middle attack. [20]

1.7.2 Hash Functions

A hash function is a one-way function h, which can quickly compute a digest (hash value) of any given input x (message), in such way that it is practically impossible to discover the value of message if only its digest is known. Input can be of any length, but each hash function returns a fixed amount of bites, this is shown in Figure 1.12. Thus, the digest can be looked at as a message fingerprint. [26]

Furthermore, the hash function should be sensitive to the changes in the input, so that even a small change in input, results in an entirely different fingerprint. If there are two different inputs x and the hash function returns the same fingerprint for both inputs, then it is called collision. Some of the oldest and most widely used hash functions, such as MD5 or SHA-1 have both discovered collisions and their use is discouraged. [26]

1.7.3 Passwords

Compromised databases are a big problem. If an attacker gains access into the database, the can potentially exploit user credentials if they are stored unprotected in plaintext. Hash

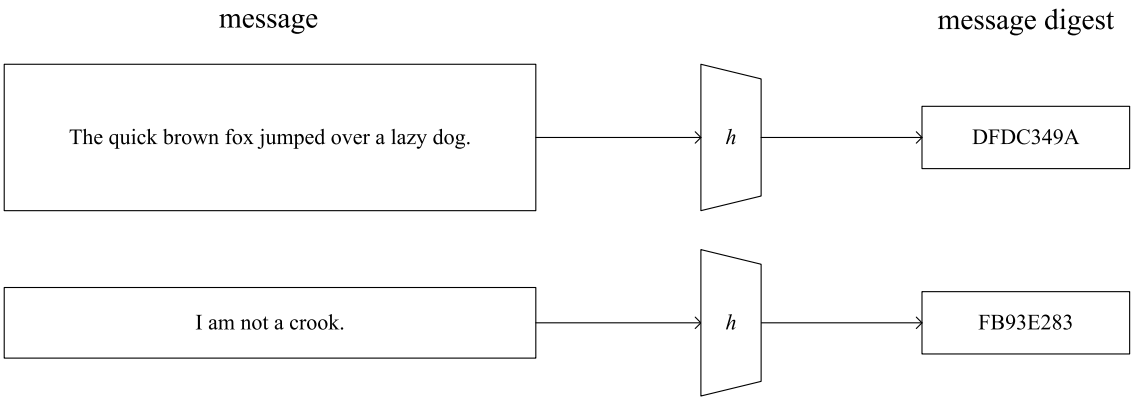


Figure 1.12: General behaviour of hash function (Source: based on [26])

functions should be used to store only a password hash. Since hash functions are sensitive to changes in their inputs, if someone tries to login with another password it should not match with the stored hash. However, a hash function which is considered secure - it has zero collisions - should be used. [3]

A password hashing does not solve every problem with password storing. What if two users have the same password? Their hashes would be identical. A salting mechanism should be implemented in order to avoid this scenario. Salt is a random string of at least 32 bytes, which is prepended or appended (depends on implementation) to a plaintext password. This string is known as a salted password. The salted password is then passed into a hash function. A password hash and salt are then stored in the authentication database. This process is displayed in Figure 1.13. [3]

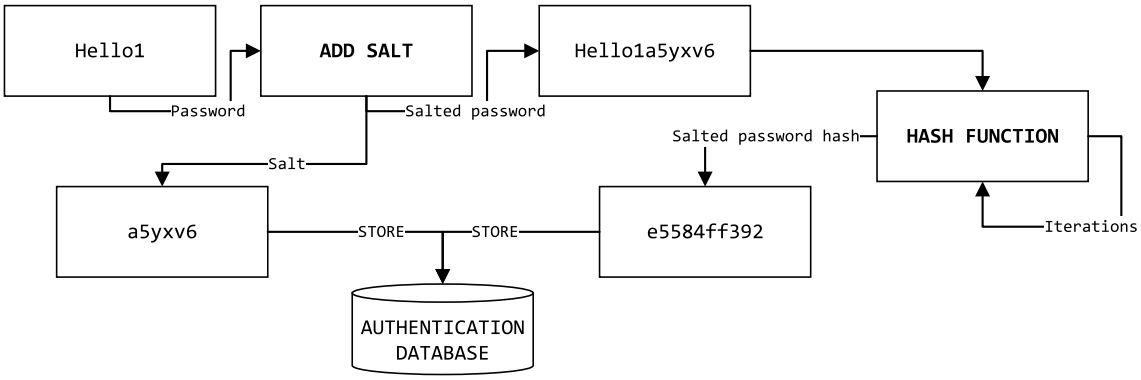


Figure 1.13: Application of salt and hashing a password (Source: based on [3])

If the password is salted before it is hashed, then even if two users use the same password, their hash would be different. Moreover, if the database would be compromised

the attacker would have access to only a table of hashes and salts. And since every password is salted, guessing the passwords by using rainbow tables would be inefficient.

1.7.4 HMAC

Hash-based message authentication code (HMAC) is a cryptographic algorithm that computes a complex function of a message (data) and a secret key. The resulting code can be used to protect the message authenticity. The basic principle is that no one should be able to compute the HMAC value without knowing the key. Therefore, after receiving a message with HMAC and knowing the secret key used to generate the HMAC value we can verify the authenticity of the message. An example of the message exchange and verification is shown in Figure 1.14. [27][2]

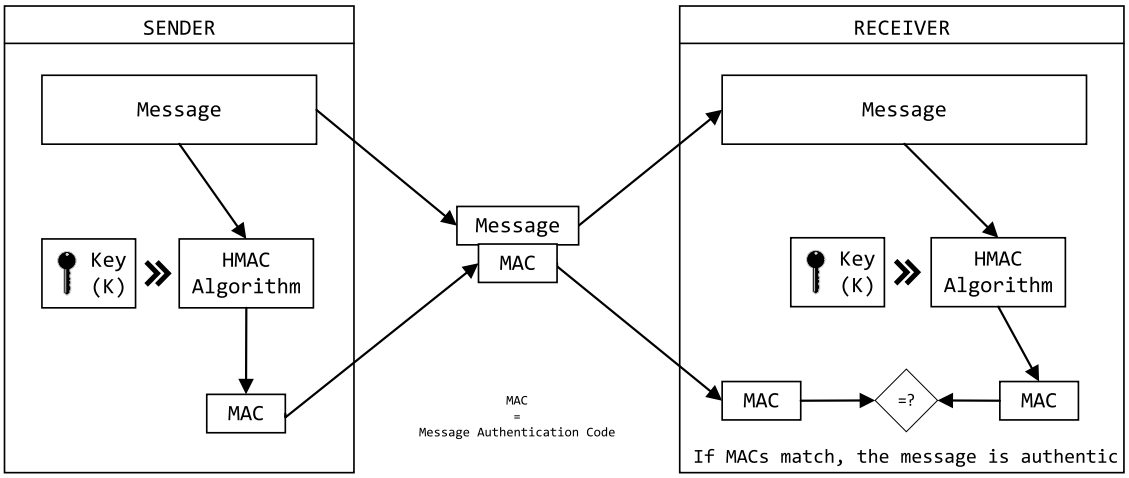


Figure 1.14: Message exchange and verification process (Source: based on [2])

1.7.5 Two-Factor Authentication (2FA)

„This term indicates user authentication where two factors are involved in the process: something the user knows (like a static password) and something the user possesses, like a token or a mobile phone used for one-time passwords.“ [18]

Implementing this method in the user authentication process makes it substantially harder for an attacker to gain access to the system. It requires that the user enters an OTP which is delivered or generated by the user. The most common method of OTP delivery is by an SMS. This, however, requires knowledge of the user's number before an authentification process starts. If this is not viable, one can also get an OTP from an

app installed on a mobile phone. These applications are often called „authenticators“, and once set up, they generate OTPs based on known secret and current time. All codes are only valid for a set period of time, but this depends on the server settings.

1.7.6 Content Security Policy (CSP)

Various kinds of a script and content injection are one of the biggest threats that developers need to think about while developing applications. Because browsers trust any content coming from the specified origin by default, they also trust any malicious scripts or content that was injected into the page. Content Security Policy (CSP) effectively creates a whitelist of origins from which a browser is allowed to load scripts or content for the specified origin. The policy appears as a series of headers sent back with a response from a web server. Origins can be whitelisted on an abstract level, there is no need to provide an extensive list of allowed origins but it is possible to deny the loading of every script which does not originate on the specified domain - this policy restriction can be seen in Figure 1.15 as an example. [22]

```
Content-Security-Policy: script-src 'self'
```

Figure 1.15: CSP header allowing only scripts embedded in page (Source: original work)

1.7.7 Attack Vectors

This section describes common attack vectors and weaknesses according to the list of the most common application attack vectors ([25]) created by the Open Web Application Security Project (OWASP).

Attackers can use many different paths through applications, finding and exploiting security weaknesses and gaining control or access to the application. An abstract example of an application attack is shown in Figure 1.16.

Denial of Service

„Denial of service is a type of attack that aims to prevent normal communication with a resource either by disabling the resource itself or by disabling the infrastructure device providing connectivity to it.“ [24]

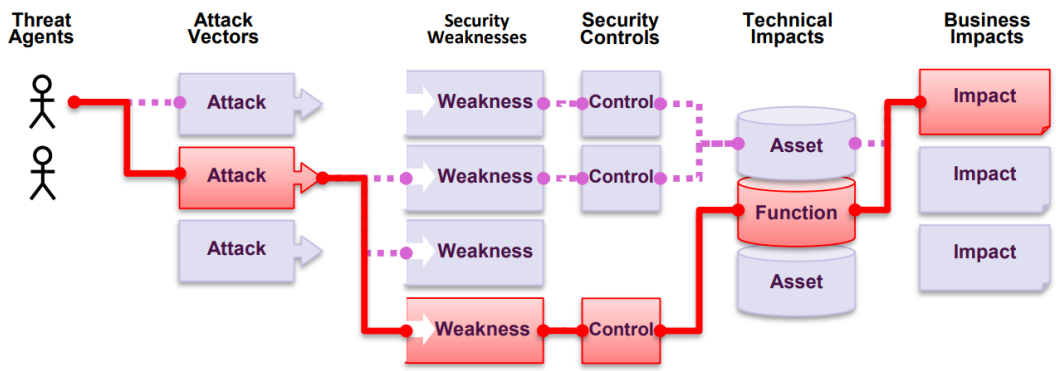


Figure 1.16: Example of an application attack path (Source: [25])

Denial of service can be achieved by flooding the victim with so much traffic that the victim can not handle additional requests. This is done either by sending a large amount of useless data which are masked to look like genuine user traffic or by sending only network packets without any data. [24]

SQL Injection

„SQL injection is a code injection technique commonly used to attack websites in which the attacker inserts SQL characters or keywords into a SQL statement via unrestricted user input parameters to change the intended query's logic.“ [14]

Not sanitising and insufficient input validation are rendering websites vulnerable to SQL injection. Web programming languages, such as PHP, provide many different methods for executing SQL queries on a database. However, these methods are often misused and introduce weaknesses into the applications. The most common misuse of these methods is dynamic query building, which is multiple strings concatenated into one and then executing it without any checks or sanitisation. [14] According to the Open Web Application Security Project (OWASP) injection was the most widespread website vulnerability in 2017. [25]

The most common defence strategy is defensive coding, but this has to be thought of in the early stage of system development. Otherwise, it is costly to rewrite and secure all database queries. If done correctly, prepared statements and stored procedures are the most effective defence strategy, as arbitrary code cannot be injected and executed. [14]

Cross-Site Scripting (XSS)

„An XSS attack involves injecting malicious script into a trusted website that executes on a visitor's browser without the visitor's knowledge and thereby enables the attacker to access sensitive user data, such as session tokens and cookies stored on the browser.“ [31] The XSS attack was ranked as the 6th most popular in the year 2017 by the OWASP community. [25]

Cross-site request forgery (CSRF)

„By launching an successful CSRF attack against a user, an attacker is able to initiate arbitrary HTTP request from that user to the vulnerable web application. Thus, if the victim is authenticated, a successful CSRF attack effectively bypasses the underlying authentication mechanism.“ [12]

Man-in-the-middle

„During this type of attack, two parties are communicating with one another and a third party inserts itself into the conversation and attempts to alter or eavesdrop on the communications.“ [24]

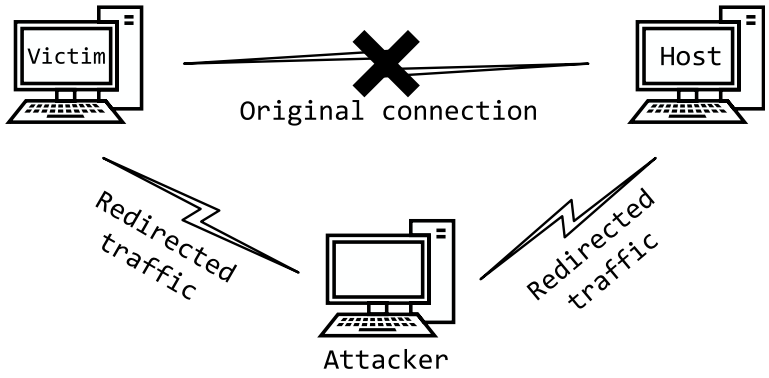


Figure 1.17: Man-in-the-middle attack (Source: based on [24])

Once the attacker is placed between the two parties, the attacker is able to passively monitor the communication or insert malicious packets in both directions. The attacker is effectively forwarding the traffic it intercepts so the two parties do not know that the attack is happening. [24]

1.8 Analytical Methods

These are methods which can be used to analyse a company and create a solid basis for development of its strategy.

1.8.1 Porter's Competitive Forces

Porter's model of competitive forces is an analytical method of examining competition of a given company. There are five areas which the competition - existing or possible - is examined. [16]

Competition in the Industry

Includes competition in a company's own industry or field of operations. A company which specialises in software development is a competitor with another company which develops software, even if one develops computer games and the other management tools. [16]

Potential of New Entrants Into an Industry

Examines how difficult it is for a competitor to enter the company's market. An industry with strong barriers is more attractive for companies as they can charge higher prices. [16]

Power of Suppliers

Examines how much can suppliers manipulate the price of their inputs which are necessary for company production. The more distinct suppliers a company has, the better position is the company in. [16]

Power of Customers

Examines the power of customers to manipulate the price of a product. If a company has a small customer base it is more susceptible to bargaining from customers. A company that has many smaller customers has more freedom in its price configuration. [16]

Threat of Substitutes

Measures how easily is the product of a company substituted for another one. If a product has a close substitute, it is generally very similar and the company has to provide additional value in order to win over a customer. [16]

1.8.2 SWOT Analysis

SWOT analysis is one of the most common analytical methods. SWOT is an acronym consisting of the factors that it represents - Strengths, Weaknesses, Opportunities and Threats. SWOT is created by analysing outputs of other analyses and company evaluations. The SWOT analysis compares strengths and weaknesses with identified opportunities and threats and serves as a starting point for defining strategies in a company. [9]

	Helpful (to achieving the objective)	**Harmful** (to achieving the objective)
Internal origin (attributes of the organisation)	Strenghts	Weaknesses
External origin (attributes of the environment)	Opportunities	Threats

Table 1.2: SWOT analysis structure (Source: based on [9])

Internal Origin

These are the factors which are inside of a company, in other words, a company has influence over these factors. The goal is to pinpoint the strong and weak sides of a company. [9]

Strengths - factors that are helpful in strengthening of company's position on the market. They are indicative of areas in which the company is ahead of its competitors. These factors include mainly knowledge, skills, resources, achievements. [9]

Weaknesses - include those areas in which a company lags behind the competitors. High costs, ineffective marketing or remote location could be classified as a weakness. Weaknesses of one company are typically strengths of the other. [9]

External Origin

These are factors that are present in the environment around a company, either as a opportunities or threats. [9]

Opportunities - factors that can bring benefits to a company if they are taken. Opportunities include technological advancements, new trends, changes in regulations and legislative. [9]

Threats - factors that could threaten a company. These include decreased demand, or tilting economic stability of a company. Typical threats are activities of competitors, changes in regulations and legislative or nature disasters. [9]

2 PROBLEM ANALYSIS AND CURRENT SITUATION

In this chapter, I will describe the company and its area of operation. The problem is described and analysed here and from its description the requirements for the system are derived.

2.1 Company Profile

Figure 2.1: Logo of the company (Source: company website)

Business name: Adam Baliak, madea systems

Identification number: 50 888 951

VAT Number: 10 857 073 81

Address: Ludvíka Svobodu 2689/80, 058 01 Poprad, Slovakia

Established: 2017

Field of operations: software development

The company established its trade as a contractor company in 2017 under the name Adam Baliak, madea systems. The company itself has many business partnerships, through which the company acquires projects and distributes workloads. The company's main field of operations is software development, specifically the development of web-based information systems. The systems include but are not limited to e-shops, WordPress sites and various extensions. The company also participated in the development of a custom CRM solution. In the year 2018, the company took part in more than 20 different projects. The company also sells web and email hosting and provides support to its clients. The most significant project was an e-shop built which contained more than 25 000 products in each of the three different language mutations.

2.1.1 Hardware

The company has two servers currently. The main server is hosted in a data centre in Brno and the other one in Prague. The server located in Brno hosts most of the company online infrastructure and also serves as a development environment. The other one serves primarily as deployment and backup server.

2.1.2 Software

Both of the company servers are running Ubuntu 18.04 LTS (Bionic Beaver). The web hosting stack consists of Nginx as its core element and atop of that its running PHP-FPM in version 5.3, 7.1 and 7.2. Jenkins[1] is used as a deployment tool. Source versioning is hosted at GitHub[2]. The company infrastructure consists of Invoice Ninja[3] used to create invoices and keeping track of clients, projects and spent hours, also a cloud storage NextCloud[4] and a modified version of Roundcube[5] webmail. Everything is self-hosted on a private company server because of the security concerns.

2.2 Products

The company itself specialises on the development of information systems. Rather recently, the focus shifted on delivering programs with specialised functionality - known as plugins - to clients. These plugins are in other words an extension of systems. Clients who expressed interest in buying these products are mostly owners of starting e-commerce sites. They would hire a contractor to build the site for them with some kind of open-source solution. However, these solutions tend to have limited integrations, let's say with payment gateways for example, and therefore the client needs to extend their system.

2.2.1 Target Audience

The target audience of the company is small to medium-sized e-commerce merchant who wants to extend their site functionality.

[1]Jenkins – an open source automation server, `https://jenkins.io`
[2]GitHub – a web-based hosting service for version control using Git, `https://github.com`
[3]Invoice Ninja – open-source online invoicing app, `https://invoiceninja.org`
[4]NextCloud – an open source, self-hosted file share platform, `https://nextcloud.com`
[5]Roundcube – free and open source webmail software, `https://roundcube.net`

2.3 The Problem

Like any other customer, the company's target audience wants to save money wherever possible. The customers are often buying plugins and reusing them on their other sites or worse - they download pirated versions of these plugins which often contain security holes.

So how can we motivate the customer to buy company products, and how can the company prevent the reuse of these products? First of all, the customer needs to understand what benefits are accompanying the company products.
The customer benefits include:

- extension of functionality
- secure source-code
- periodical automatic updates
- customer support

Still, the customer will reuse the product if they have a chance. To prevent this the company wants to include these features in its products and the Licence Management System:

- product activation feature -> limited functionality until activated
- periodic check to see if updates are available on the server
- require reactivation if the system domain changed
- the system will authenticate request based on origin
- licences will be issued only for one domain, but can be transferred

2.4 Porter's Competitive Forces

The goal of analysing competitive forces is to find a position for the company in which it will be most prosperous.

Competition in the Industry

In the field of software development - mainly web applications - there is a big number of companies providing these services. However, there is only one company which has developed products similar to our company.

Potential of New Entrants Into an Industry

The entry barrier is in this industry rather small, but customers do not trust newly established companies. However, the company has to develop numerous products if it wants to satisfy a broader spectrum of customers and that is time-consuming. Moreover, they would need a system for managing licences or some other solutions.

Power of Suppliers

The industry in which the company is operating does not require a suppliers as the company is creating value by itself.

Power of Customers

The company has moderate number of customers and accepts new orders on a per-project basis. However, with each new project there is a possibility of up-selling company's products. Typical customers of the company are merchants founding their e-commerce sites.

Threat of Substitutes

Number of substitutes for company's products is low as there is only one other company which develops and maintains them. This company, as it is currently alone in this field, has expensive products. There will be a space to manipulate the prices to make the products more appealing to customers.

2.5 SWOT Analysis

	Helpful (to achieving the objective)	**Harmful** (to achieving the objective)
Internal origin (attributes of the organisation)	contact with clients, knowledge of the problem, experience with client systems, strict permission scheme, well-defined functionality, internally developed system	limited resources, small infrastructure, high cost of the system
External origin (attributes of the environment)	small competition, company partners, small number of substitutes	software piracy, security threats, problem requires custom solution

Table 2.1: SWOT analysis (Source: original work)

The company has good knowledge of the problem it is trying to solve, good contacts and experience with client systems. Amongst other benefits, the system can be developed internally, which will help with maintenance in the future. On the other hand, since this is a custom solution, designed to fulfil specific function it is more expensive than some other ready-made solutions which would, however, require heavy customizations in order to meet company requirements.

The security of the system has to be taken seriously as the system will be publicly accessible and cannot fall as a victim to numerous security threats. Other harmful factors include the limited resources of the company. Having more development resources would strengthen the company's position on the market.

2.6 Implementation Strategy

Before the work on the system can start both parties have to agree on an implementation strategy. For the development of the Licence Management System, the following implementation strategy was chosen.

1. Analyse the system requirements
2. Propose a concept of the system, see if meets all of the criteria
3. Implement the proposed solution
4. Run tests in the development environment to validate the functionality
5. Deliver the system

2.7 System Requirements

The information system should provide a platform on which licences for digital products can be easily managed. Alongside the licence management, it should allow for management of product for which a licence can be issued and also for the system users.

2.7.1 Functionality

Products

A digital product can be any software that extends the functionality of another software, and or provides functionality on its own. The company is currently developing WordPress plugins and themes, so for now supporting those products is the priority. There may be additional types of products sold in the future, so the system should be easily extendable.

Licences

A digital licence is bought by a customer which can afterwards activate the selected product and receive automatic updates. One licence shall allow for unlimited activations, but only on one domain which is provided by the customer when ordering selected product. The system will enable editing licence properties afterwards by a user with administrator privileges.

Users

There will be two main groups of system users with different permissions. One is an administrator group who will manage licences, products and users. Moreover, administrators can see performance statistics of the individual products. The second group is for resellers who are selling company's products via their own channels and generate licence keys for their users for which they are billed afterwards. These users should be only able to generate licence keys for products which are assigned to them by the administrator. They can not edit properties of the licence keys they have issued, nor they can perform any other action on the system.

A specific group of users are clients. They have activated products running on their systems which have the ability to ask for updates. These requests will be satisfied if they are coming from the domain registered in their licence.

Statistics

Since the company has limited resources, the development and support of the products have to be focused. Statistics should provide insight into the number of users, activations and current usage of the individual product versions running on the client systems.

2.7.2 Usability

The system should be easy to navigate, simple and should use a typical layout of three components - menu on the left side, navbar with user options on the top and big content window. It should be responsive as the usage on small screen resolutions is expected.

2.7.3 Security

The system will be used to generate income for the company and therefore ensuring the data integrity is crucial. Furthermore, the system will contain sensitive information which shall not be made available to the third party. The system shall also satisfy only those update requests which are deemed valid according to the licence details and prevent forging of the parameters used in these requests.

2.8 Permission Scheme

Based on the given requirements, the following permission scheme is proposed. It defines roles which users can play while interacting with the system.

Permission	User role		
	Administrator	**Reseller**	**Client**
General			
Access API interface	no	no	yes
Access web interface	yes	yes	no
Products			
List all products	yes	no	no
Add new product	yes	no	no
Show product properties	yes	yes	no
Edit product properties	yes	no	no
Show product statistics	yes	no	no
Licences			
List all licence keys	yes	no	no
List own licence keys	yes	yes	no
Generate a new licence key	yes	yes	no
Edit licence key properties	yes	no	no
Verify licence validity	yes	yes	no
Users			
List all users	yes	no	no
Add new user	yes	no	no
Edit user profile and role	yes	no	no
Edit own profile	yes	yes	no
Assign products to users	yes	no	no

Table 2.2: User permission breakdown (Source: original work)

Administrator

Administrator is a person with full access to the whole system. He can manage other users, add and edit new products. These users will be company employees who are responsible for managing licences.

Reseller

Reseller is a person who was authorised to sell company's digital products and is allowed to use the system. This person is able to browse web interface, display products that are available to sell and generate licence keys for such products.

Client

Client is a person which has installed company's products on their system and is allowed to request updates. Since the updates are automatic, there is no need for the client to access the web interface, or otherwise interact with the system.

2.9 Security Threats

Security Threats	Origin		Intention			Factor		Threatened Asset					Security Attributes			Motivation				
	External	Internal	Random	Intentional	Unintentional	Nature	Human factor	Hardware	Network connectivity	Data at rest	Data in motion	Users	Availability	Confidentiality	Integrity	Financial gain	Competitive superiority	Proof of skills	Revenge	Non-compliance
DDoS attack	X			X			X	X	X	X			X			X	X	X		
Cross-site scripting	X			X			X			X		X			X	X				
SQL Injections	X			X			X			X		X		X	X	X	X	X		
Cross-site request forgery	X			X			X			X					X	X	X	X		
Man-in-the-middle attack	X			X			X				X	X		X	X	X		X		
Broken authentication	X			X			X			X		X		X		X	X			
Weak password		X			X		X			X		X		X						X
System feature bug	X				X		X			X		X			X					
System security bug	X				X		X			X		X		X						
Security misconfiguration		X			X		X	X	X	X			X	X	X					
Server failure		X	X			X		X	X	X			X							
System misuse		X			X		X			X					X					X
Unsatisfied employee		X		X			X			X			X	X	X				X	
Nature disaster	X		X			X		X	X			X	X		X					

Table 2.3: Overview of possible threats to the system (Source: original work)

Table 2.3 shows an overview of possible security threats which could affect the correct functioning of the system. These threats mentioned are amongst the most popular vulnerabilities of information systems. The system needs to implement measures in order to prevent threats like cross-site scripting and cross-site request forgery. Processes in the system should adhere to the essential security requirements specified in Section 1.7.

Additional emphasis should be given o proper introduction of the system to company employees in order to prevent system misuse.

Some of the measures have to be taken at a server level like proper configuration of firewalls to prevent DDoS attack or updated security configuration of the server. For example, natural disaster threat could be mitigated by having the system run in a distributed manner but this type of architecture requires complex configuration and consideration and is out of the scope of this system design. However, the system should be designed in a way that it allows for this type of configuration in the future.

2.10 Problem Analysis Evaluation

The company finds itself in a unique position in the industry. Opportunity to enter a new field of digital products is presenting itself and the Licence Management System will allow the company to build upon this opportunity. As the analysis of competitive forces shows, the competition in this field is currently very low so the company will have space to configure prices to be appealing to the customers and still support the development of new products.

If the system can be designed and implemented according to the company requirements there is a place for it in the company portfolio. However, some requirements will require unique approach and implementation.

3 PROPOSALS AND CONTRIBUTION OF SUGGESTED SOLUTIONS

In this chapter, I will focus on the design of the Licence Management System. First, I will address the company requirements and model a system architecture and a database accordingly. I will afterwards implement and test the system on the company's server.

3.1 Proposed Solution

According to the company requirements, the system will need to fulfil several functions. The primary function of the system will be the web interface which provides management functionality to the authorized users. This interface will be developed in HTML and styled with CSS, user experience should be improved by using asynchronous transfers where possible, dynamic functionality and business logic will be written in PHP. Data will be stored in a relational database, using the open-source MySQL database engine. There shall be only a limited amount of customer data stored at rest and they should be protected at the server level. The system shall communicate with its user via a secured HTTPS connected to protect data in motion. The update functionality will be provided by an open-source update server which will be integrated into the system which will authorize incoming update requests with the system. This updating functionality will be also built into the company products, however, that implementation is out of the scope of this thesis.

The security threats described in section 2.9 shall be addressed if they can be prevented by the system implementation. The company is responsible for securing the environment in which the system will be running - the servers.

3.2 System Architecture

To give the system a perspective, I have used a C4 model to create a system context and a container diagram. This model is described in section 1.3.3. The system was designed as a n-Tier system with frontend and backend components communicating together.

3.2.1 System Context View

This diagram shows our information system - Licence Management System (light blue box) and how it interacts with external systems. It also shows which roles can users play relative to the system. Furthermore, use cases can be seen here too.

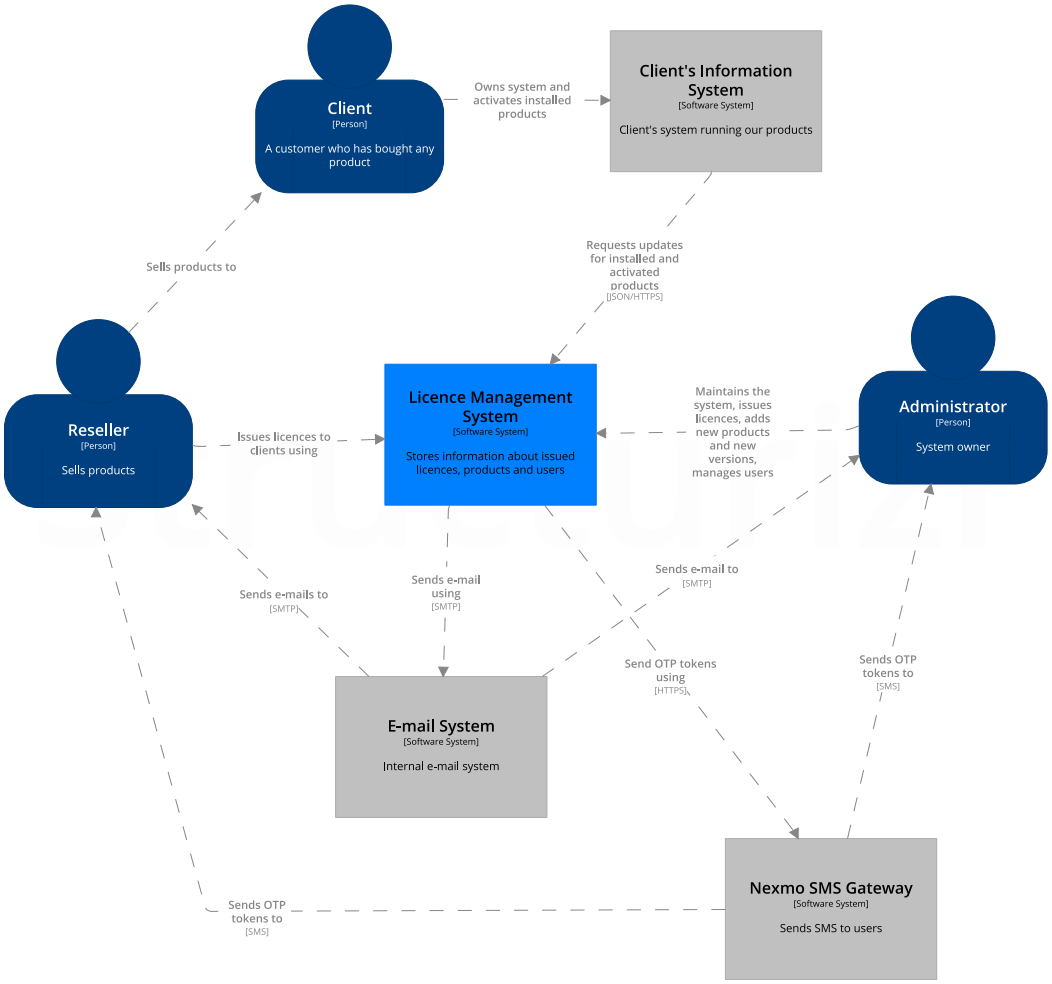


Figure 3.1: System context diagram (Source: original work)

The system directly interacts only with an e-mail server, to which it connects via the SMTP protocol because we only need the sending capabilities. The e-mail server can be running on the same server as the system or it can be a remote server. The other service that we are using is an SMS gateway service to which we are making API calls containing data serialized in a JSON format, this is described more thoroughly in section 1.2.4.

3.2.2 System Container View

This diagram breaks down the Licence Management System and shows its internal service containers and how they interact with each other.

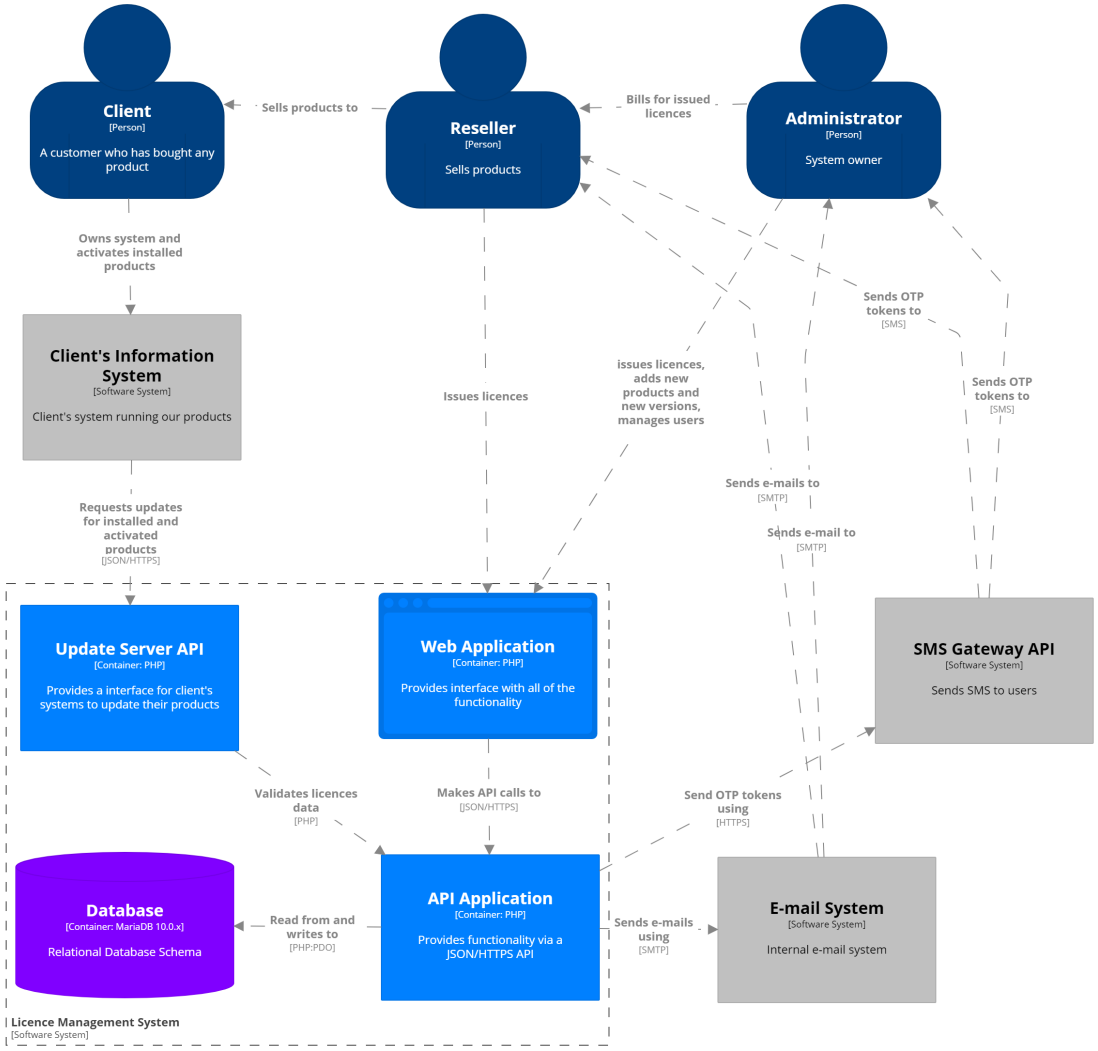


Figure 3.2: System container diagram (Source: original work)

The Licence Management Systems consists primarily of four service containers and two entry points, through which the system can be accessed. Administrators and resellers interact with the Web Application, which then loads and changes data using the API Application. Data are stored in the Database, and they can be accessed only through the API Application.

Products installed on the client's system are configured to use the Update Server API, where they can check if any applicable updates are available. This communication is subject to the verification process, which detailed description is available in section 3.6.4.

3.3 Database Model

An entity relationship diagram described in section 1.3.2 was used to effectively describe the database model. In this model, we can see relationships between entities stored in a database. As you can see in the figure 3.3, we are using only one table for users as the only thing that differs between users are the access rights (permissions) and these are defined in a different table. There is a limit of only one email or phone number per user, otherwise, we would store them in a separate table as well. We also do not need to store user address or any other information, as these are stored in the company's accounting system.

The whole model was designed with data consistency in mind. Therefore, we are frequently using foreign keys with predefined constraints. These restrictions are beneficial for keeping the database clean and not storing useless data. For example, when the company decides to end a partnership with a reseller, information like reseller's phone number, email, access to the system are deleted automatically, with the help of `ON DELETE CASCADE` constraint. There are few defined scenarios when this constraint is in action.

On the other hand, this behaviour would be undesired for the number tables. For example, the table containing information about various platforms or languages. This is because we would lose all of the products for said platform or given language. The constraint is therefore set to the `SET DEFAULT`, so when we delete a row the value will be replaced with the default value for that column.

The last integrity constraint that is used in the database is the `RESTRICT` option, which prevents the deletion or modification of the record when there are relations in the table referencing that foreign key.

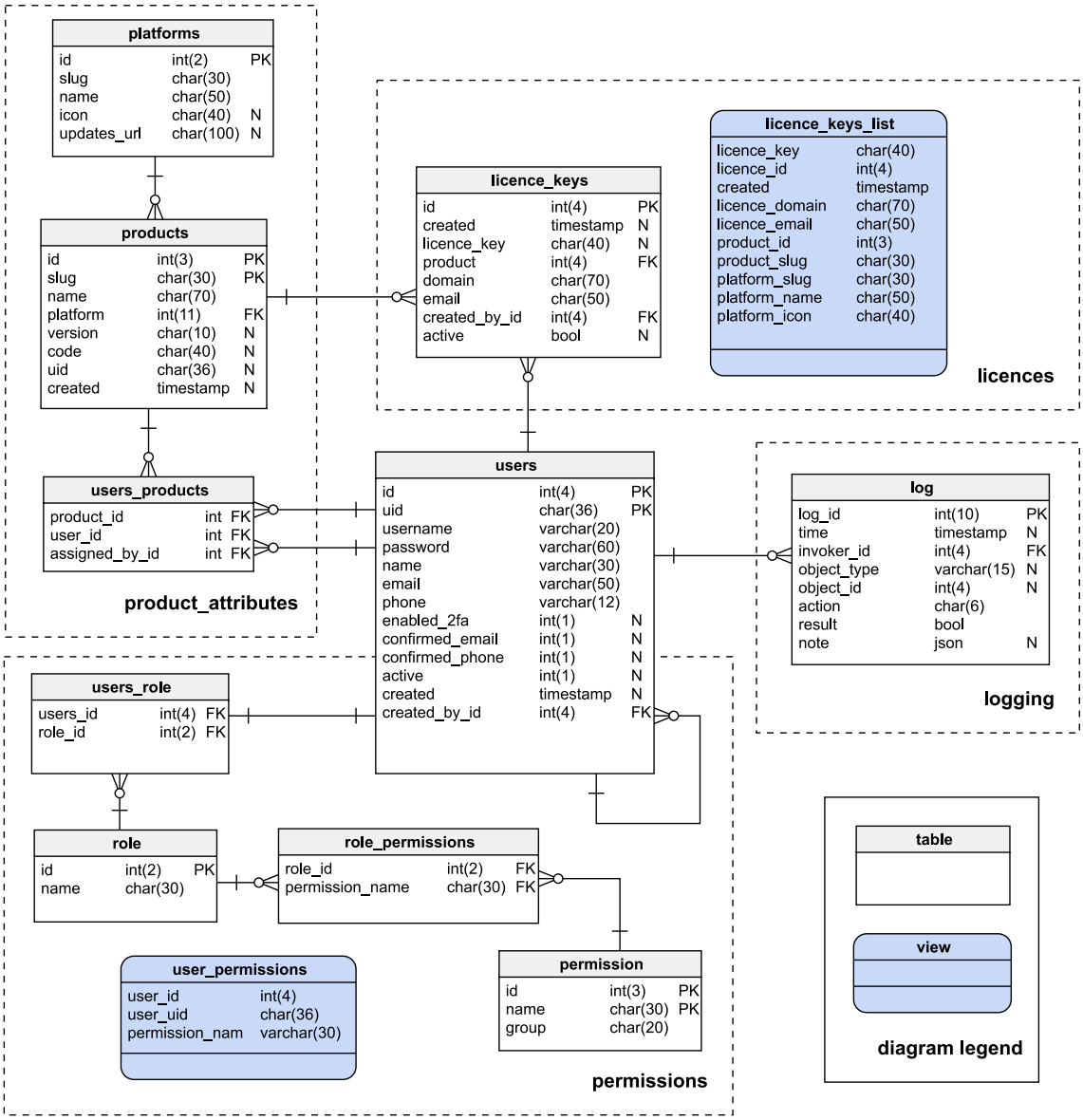


Figure 3.3: ER diagram (Source: original work)

3.4 User Interface

The user interface was designed with two core values in mind. It should be responsive and functional. I followed a traditional layout of a big menu on the left side and header with options in the top section. With these requirements, I did search for a complete template which fulfilled all of my requirements. This is a commonly used technique in web development, as it saves time which would be otherwise spent on design and reinventing the wheel. Furthermore, this is an application which will be used only by the

company's employees and approved resellers, and therefore the requirements were reduced mainly to the functional aspect of the design.

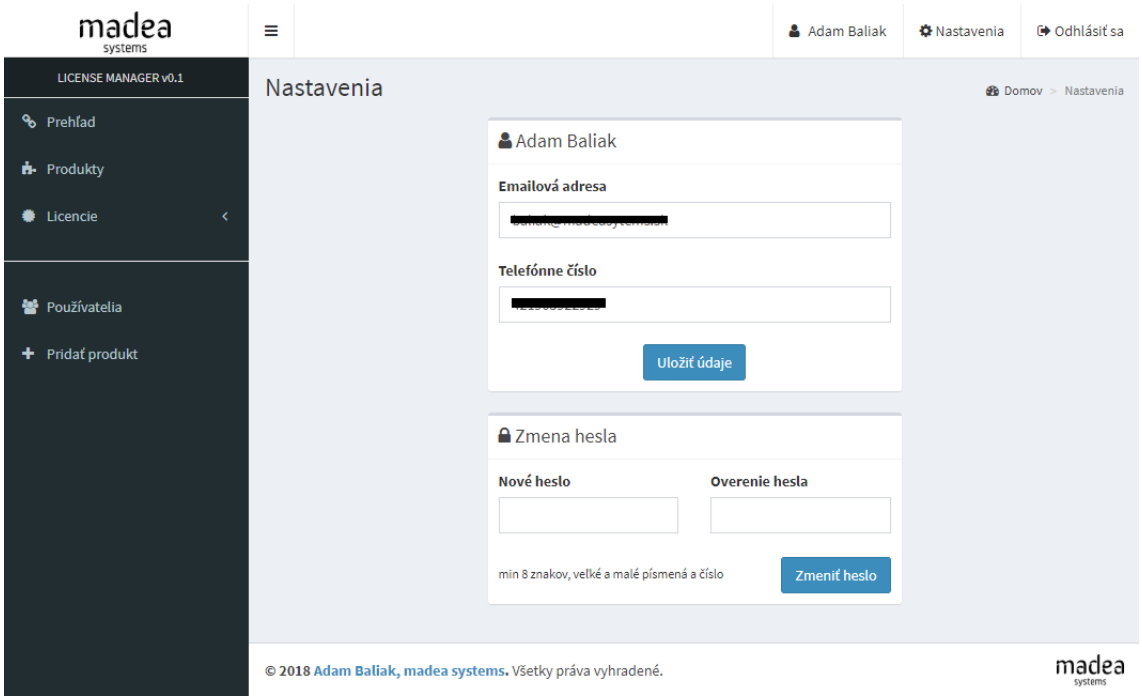


Figure 3.4: Design of the user interface - big screen version (Source: original work)

The whole user interface is built with Bootstrap 4, as this is more-or-less an industry standard nowadays for building responsive web applications. The template which I did use implements breakpoints in CSS stylesheets, so every rule applies only to the specified screen resolution. By choosing this approach, we reduced the number of stylesheets needed to load when presenting the application, because the source code used on both versions is identical. Breakpoints are set to cover devices with large screens (larger than 1366px) typically laptops and computers, medium screens (width is more than 768px, but less than 1366px) and small screens (less than 768px) aimed primarily to cover mobile phone screens. Differences between large and small screen versions can be seen on figures 3.4 and 3.5 respectively.

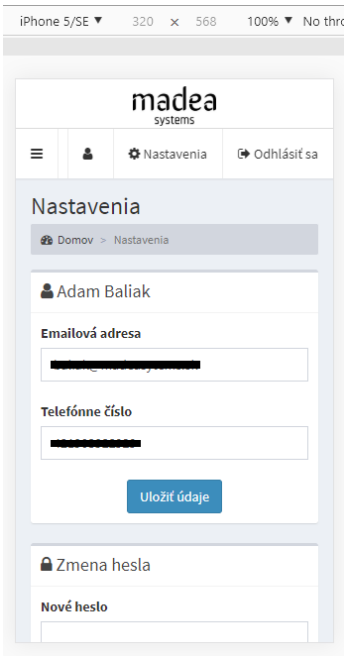


Figure 3.5: Design of the user interface - small screen version (Source: original work)

3.5 Application Security

This section describes which security measures were implemented in order to enhance application security. All of the mentioned practices are described in Chapter 1 Theoretical Background. These measures are more or less industry standard when it comes to the application protecting.

3.5.1 Database

The database is the most valuable application asset, and therefore extra measures had to be taken in order to protect it adequately. One of the most simple, yet most effective measure was to deny access to all of the tables completely and to only allow data access and manipulation through predefined procedures, functions and views. This measure was implemented by limiting permissions for the user used by the application.

Furthermore, this kind of use restriction enables us to implement a thorough logging process. Every action which is done by calling a procedure or a function is logged into a separate table which can be afterwards audited by a system administrator.

3.5.2 User Authentication

With the increasing trend of password reuse, the threat of user impersonation had to be addressed. Every account which is created in the system has by default enabled two-factor authentication. Therefore it is required for the users to provide a telephone number, on which they will receive a one time password each time they try to access the system. This security measure is described in section 1.7.5.

An SMS[1] gateway service was chosen and integrated into the system. Each time a user tries to log into the system, a one-time password (a six digit random number) is generated, stored in the PHP session variable and then an API call is made to the gateway service, requesting an SMS message containing the one-time password to be sent to the phone number provided by the user. The one-time password is valid only for one minute. If the user fails to provide this password in time, then the authentication process fails, and the user has to start over.

3.5.3 Password Storage

Passwords are not stored in the database, instead, their hashes are stored. Passwords are also salted before the hashing process. This method was implemented as adds all of the benefits as described in section 1.7.3. As a hashing algorithm was chosen Blowfish since it is secure, has no discovered collisions and the hash produced by this algorithm contains the salt which was used to „salt“ the password.

3.5.4 Authentication Bypass

In order to protect the application from the authentication bypass, every request which is received on the server has its headers checked. If the domain in the referrer field is not on the whitelist, then the request is denied. This is however not a complete solution, as these headers can be sent empty or altered to look genuine like the request was sent from the application domain. Therefore a nonce system had to be implemented. Every form which is displayed to the user contains a nonce (token), which is bound to the user and can only be used once. So if the attacker wants to perform this type of attack, this nonce has to be guessed and included in the request. A CSP policy was also adopted.

[1] SMS – short message service

3.6 Functionality Implementation

This section describes how was the specific functionality designed and implemented. Behind every functional decision there has to be a reasoning. At first I did validate that proposed design fulfils all of the expected criteria and implemented it afterwards.

3.6.1 Licence Keys

Licence keys are generated by the database after a successful request. Licence keys are in the form of UUID version 4, as specified in the RFC4122 document [19]. Strings generated by this algorithm have a diminutive chance of collision. It can also be safely used as a primary key in database tables. Because each licence key needs to be unique, that is why I did choose this algorithm for producing the licence keys. A licence key which was generated on the database can be seen in figure 3.6 as an example.

```
b988059a-ef68-441d-8acf-9c3d3efa6629
```

Figure 3.6: Licence key example (Source: original work)

3.6.2 Application Domain

Since this is a web application, it needs a publicly accessible address on the internet. The company owns a second-level domain madeasystems.sk. The system should be installed on its own domain, potentially a subdomain, so access to it can be more controlled. The candidates for this domain are:

- licences.madeasystems.sk
- plugins.madeasystems.sk
- lcs.madeasystems.sk

Advantages of using a third-level domain are that since the company already own a second-level domain, it will not incur any additional costs to register it.

3.6.3 Client System Fingerprint

A fingerprint request consists of a product UID which is stored in the product source code. When the request is received by the Licence Management System, the system extracts an IP address and a URL from the request headers. These parameters are passed to the fingerprinting function. This function checks the URL and tries to determine the client system domain and if it is successful it then fetches the domain IP address from the public DNS systems. Afterwards, the IP address extracted from the request is compared with the IP address of the domain. Only in the case that these addresses match, a fingerprint is produced. The fingerprint is an HMAC hash with three parameters, an IP address, client system domain, and a product UID. This fingerprint is unique to the client system which requested it and it cannot be modified or generated by the client system because of the HMAC hash properties. The HMAC hash is described in section 1.7.4. An advantage of how this process was designed is that it does not have to query the database in order to generate a fingerprint.

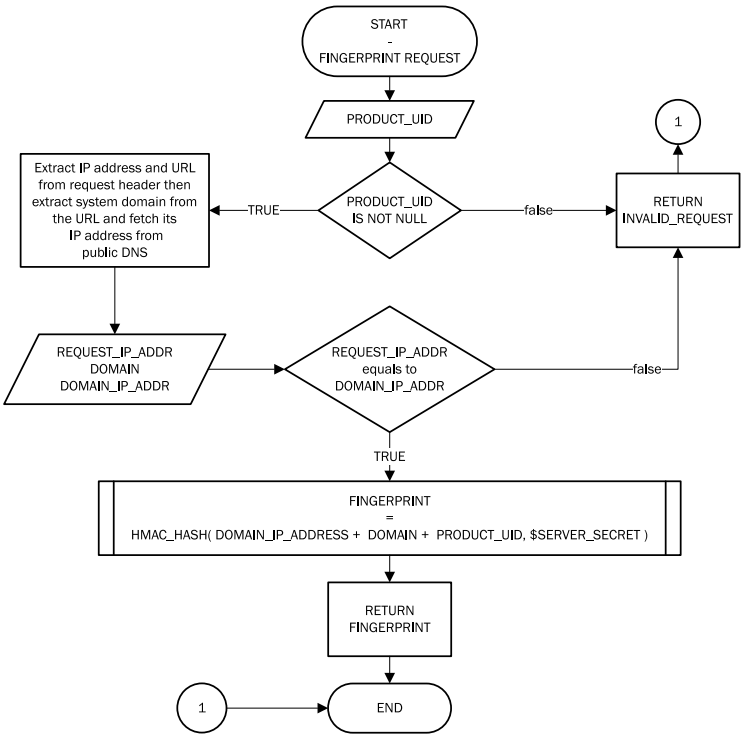


Figure 3.7: Flowchart of the fingerprint generation process (Source: original work)

3.6.4 Verification Process

Whenever a client installs a company's product on their system, in order to enable its functionality it has to be activated by sending an activation request to the Licence Management System. *The validation process of an activation request is identical to the validation of an update request.* At first, the client system has to obtain a fingerprint for the product it wants to activate. The process of obtaining a fingerprint is described in the section above (section 3.6.3). If it is successful and obtains a fingerprint it then sends an activation request which consists of a licence key and the fingerprint. The request does not have to contain a product UID or any other identifying parameter because they will be derived from the licence key attributes stored in the system. These attributes will be used to generate another fingerprint on the server which will be compared with the values stored in the fingerprint sent in the request. This fingerprint will be then compared to the fingerprint sent in the activation request. If these fingerprint match the request is deemed as valid and product will be activated or an update will be provided for it.

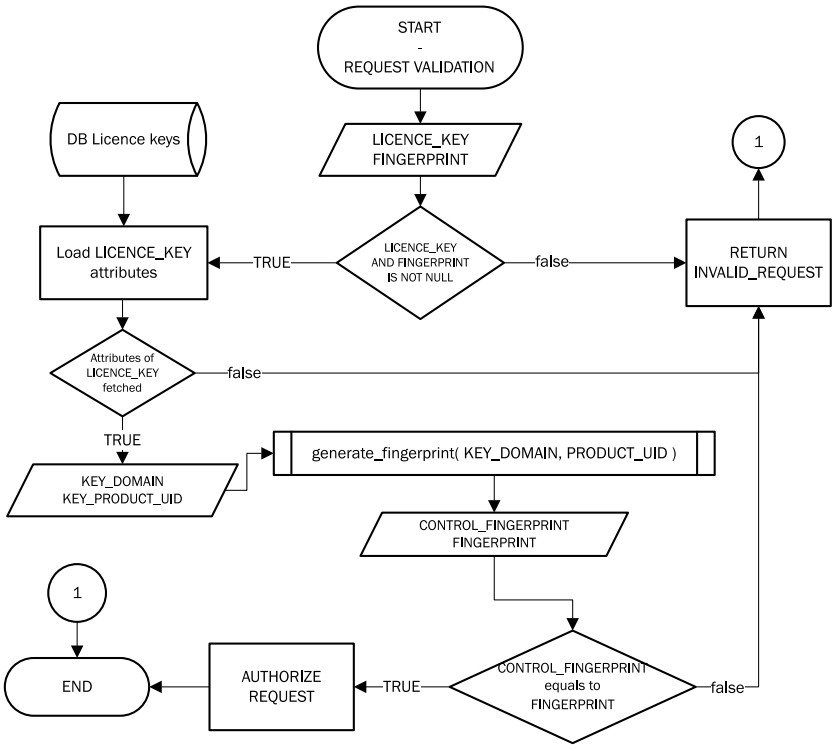


Figure 3.8: Flowchart of the request verification process (Source: original work)

3.6.5 Product Statistics

The system itself does not provide individual statistics for products. It has however integrated an update server for WordPress plugins and themes, which logs all of the update requests and holds them in a separate database. After receiving a request it checks with the License Management System if it should satisfy the request. This solution was chosen since these are the only products the company is selling currently and it also helped to speed up the system development. This update server is described in section **??**. The statistics are accessible from the list of products and include the number and versions of active installs, API calls, PHP and WordPress versions running on the client systems. These data are visualised in a form of easily readable graphs as you can see on figure 3.9.

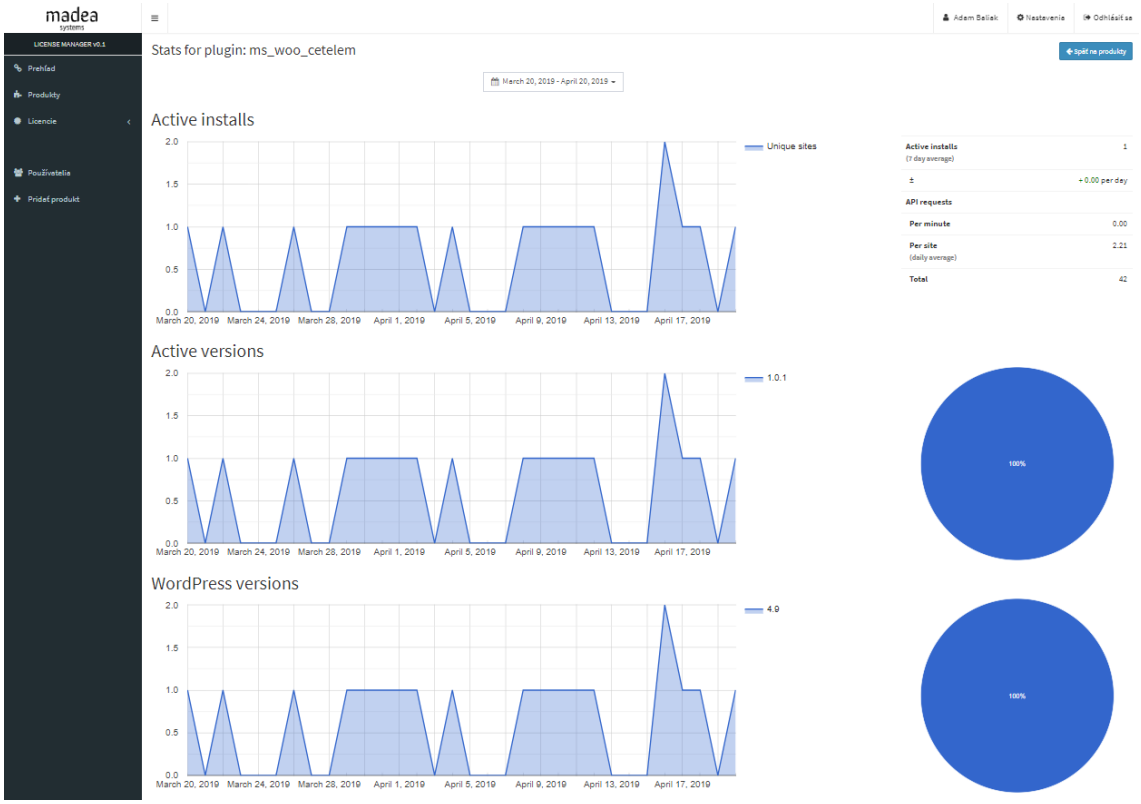


Figure 3.9: Screenshot of the product statistics page (Source: original work)

3.7 System Evaluation

The system's aim is to minimize the costs of supporting digital products developed by the company. The designed and implemented system should provide economic benefits to

the company. It should contribute to the competitiveness of the company by providing an easily extendable, simple and secure platform to manage digital products. Moreover, the usage statistic is an invaluable information source, because the company can better focus its development of individual products.

3.7.1 Commissioning the System

The system is currently in the beta stage. The verification and licencing functionality have been implemented in multiple products which are installed on client systems. The pilot stage is expected to end by the end of the summer of 2019. The system is deployed on the company server, which is hosted at a data centre in Brno with guaranteed availability.

3.7.2 System Future

The company plans to extend the system in the future. A more complex product versioning system can be created by connecting the system with the company's Git repositories. This would automate the product release process. If the number of active users will significantly increase, it will be more cost-efficient to negotiate a contract with a mobile operator to get a better rate for SMS messages, since they are used in the authorisation process.

3.7.3 Economic Evaluation

The information system described in this thesis was developed internally for the company. Modelling and designing the system took me around 15 hours, including researching the technologies needed to satisfy the system requirements. The implementation and integration of the system lasted around 90 hours. Total time spent on the development of the information system was 105 hours. The average charge for an hour of my work is 200 Kč. Therefore, the costs associated with the system development were approximately 21.000 Kč. However, since the system was developed internally these are only opportunity costs of the time I could spend working on other projects.

The system itself has relatively low hardware requirements and there is no need for the company to invest in new infrastructure. Software costs associated with the system development are zero because all of the used software was open-source, which means that

the company did not have to buy any software licences. The use of open-source software also benefits from the support that the open-source community provides.

One of the main benefits of the information system is that it fully automated the update process of the sold products. This saves employee's time which was effectively wasted on responding to the client tickets. It also saves time for the clients because they do not have to manually update products installed on their systems.

CONCLUSIONS

This bachelor thesis describes the steps which I have taken in order to design and implement the requested information system for a company specialising in information systems development. At first, I did analyse the requirements which were given on the system, then I proposed a solution and processes. When the proposed solution aligned with the strategy of the company, I started implementing the system.

I believe that the system satisfies all of the company requirements and will create value for the company and its clients. Benefits that the system brings to the company are numerous. The system acts as a management platform which enables effective management of licences and considerably reduced the costs of supporting company products.

While I was working on the design of the system I deepened my understanding of information security and principles used to develop a secure system.

In the end, however, I find it important to mention that digital products are very niche in terms of software piracy as the source code is distributed directly to the clients. Having no control over the source code once it is delivered means that with some level of programming knowledge, anyone can modify the product to circumvent the activation procedures built into the products. Therefore, the company had to motivate its clients to buy the products and not reuse them on all of their systems. The update functionality was created in a way that it requesting an update for a product which was not properly activated, or does not have a valid licence is not possible. Therefore, for the client to have a secure and updated product installed on their system, he has to buy the product licence.

BIBLIOGRAPHY

[1] BASKARADA, S.; KORONIOS, A.: Data, Information, Knowledge, Wisdom (DIKW): A Semiotic Theoretical and Empirical Exploration of the Hierarchy and its Quality Dimension. *Australasian Journal of Information Systems*. vol. 18, no. 1. 2013. ISSN 1449-8618. doi:10.3127/ajis.v18i1.748. [Online, accessed on 19.04.2019].
Retrieved from: `http://dx.doi.org/10.3127/ajis.v18i1.748`

[2] BHARGAV, A.: Security Engineer Interview Questions: What's an HMAC? 2018. [Online, accessed on 15.04.2019].
Retrieved from: `https://medium.com/@abhaybhargav/security-engineer-interview-questions-whats-an-hmac-aaf6406e5897`

[3] BLUE, J.; FUREY, E.: A Novel Approach for Protecting Legacy Authentication Databases in Consideration of GDPR. In *2018 International Symposium on Networks, Computers and Communications (ISNCC)*. June 2018. pp. 1–6. doi:10.1109/ISNCC.2018.8531022. [Online, accessed on 10.04.2019].
Retrieved from: `https://doi.org/10.1109/ISNCC.2018.8531022`

[4] BRAY, T.: The JavaScript Object Notation (JSON) Data Interchange Format. STD 90. RFC Editor. December 2017. doi:10.17487/RFC8259. [Online, accessed on 15.04.2019].
Retrieved from: `https://doi.org/10.17487/RFC8259`

[5] BROWN, S.: The C4 Model for Software Architecture. *InfoQ*. 2018. [Online, accessed on 06.04.2019].
Retrieved from: `https://www.infoq.com/articles/C4-architecture-model`

[6] BURDETT, A.; BURKHARDT, D.; CUMMING, A.; et al.: *BCS Glossary of Computing and ICT*. Swindon: BCS Learning & Development Limited. 12 edition. 2008. ISBN 9781906124007.

[7] FLANAGAN, D.: *Javascript: The Definitive Guide*. Sebastopol: O'Reilly Media, Incorporated. sixth edition. 2011. ISBN 9780596805524.

[8] FRANCISCO, J.; SADIKOV, V.: Structured Approach to Web Development. *CoRR*. vol. abs/1405.1992. 2014. `1405.1992`.
Retrieved from: `http://arxiv.org/abs/1405.1992`

[9] GRASSEOVÁ, M.: *Analýza podniku v rukou manažera : 33 nejpoužívanějších metod strategického řízení*. Brno: Computer Press. second edition. 2012. ISBN 978-80-265-0032-2.

[10] HARRINGTON, J. L.: *SQL Clearly Explained*. Saint Louis: Elsevier Science & Technology. third edition. 2003. ISBN 9780123756978.

[11] JOHANN, S.: Software Architecture for Developers. *IEEE Software*. vol. 32, no. 5. Sep. 2015: pp. 93–96. ISSN 0740-7459. doi:10.1109/MS.2015.125. [Online, accessed on 06.04.2019].
Retrieved from: `https://doi.org/10.1109/MS.2015.125`

[12] JOVANOVIC, N.; KIRDA, E.; KRUEGEL, C.: Preventing Cross Site Request Forgery Attacks. In *2006 Securecomm and Workshops*. New Jersey: IEEE. 2006. ISBN 1424404223. pp. 1–10. doi:10.1109/SECCOMW.2006.359531. [Online, accessed on 20.02.2019].
Retrieved from: `https://doi.org/10.1109/SECCOMW.2006.359531`

[13] KŘÍŽ, J.: *Databázové systémy*. Učební texty vysokých škol. Brno: Akademické nakladatelství CERM. vyd. 1. edition. 2005. ISBN 80-214-3064-8.

[14] KHIN SHAR, L.; BENG KUAN TAN, H.: Defeating SQL injection. *Computer*. vol. 46, no. 3. 2013: pp. 69–77. ISSN 00189162. doi:10.1109/MC.2012.283. [Online, accessed on 17.02.2019].
Retrieved from: `https://doi.org/10.1109/MC.2012.283`

[15] KOCH, M.; NEUWIRTH, B.: *Datové a funkční modelování*. Učební texty vysokých škol. Brno: Akademické nakladatelství CERM. vyd. 4., rozšířené edition. 2010. ISBN 978-80-214-4125-5.

[16] KORÁB, V.; REŽŇÁKOVÁ, M.; PETERKA, J.: *Podnikatelský plán*. Brno: Computer Press. vyd. 1 edition. 2007. ISBN 978-80-251-1605-0.

[17] KROENKE, D.: *Databáze*. Brno: Computer Press. first edition. 2015. ISBN 978-80-251-4352-0.

[18] LANDROCK, P.: Two-Factor Authentication. In *Encyclopedia of Cryptography and Security*, edited by H. C. A. van Tilborg. Boston, MA: Springer US. 2005. ISBN 978-0-387-23483-0. pp. 638–638. doi:10.1007/0-387-23483-7_443. [Online, accessed on 18.02.2019].
Retrieved from: `https://doi.org/10.1007/0-387-23483-7_443`

[19] LEACH, P.; MEALING, M.; SALZ, R.: A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122. RFC Editor. July 2005. doi:10.17487/RFC4122.
Retrieved from: `https://doi.org/10.17487/RFC4122`

[20] LEBLANC, J.; MESSERSCHMIDT, T.: *Identity and Data Security for Web Development*. Sebastopol: O'Reilly Media, Incorporated. first edition. 2016. ISBN 9781491937013.

[21] MANUEL, P. D.; ALGHAMDI, J.: A data-centric design for n-tier architecture. *Information Sciences*. vol. 150, no. 3. 2003: pp. 195 – 206. ISSN 0020-0255. doi:10.1016/S0020-0255(02)00377-8. [Online, accessed on 10.04.2019].
Retrieved from: `https://doi.org/10.1016/S0020-0255(02)00377-8`

[22] MUELLER, J. P.: *Security for Web Developers*. Sebastopol: O'Reilly Media, Incorporated. first edition. 2015. ISBN 9781491928646.

[23] NIXON, R.: *Learning PHP, MySQL and JavaScript*. Sebastopol: O'Reilly Media, Incorporated. fifth edition. 2018. ISBN 9781491978917.

[24] ORIYANO, S.-P.: *Cehv9 : Certified Ethical Hacker Version 9*. New York: John Wiley and Sons, Incorporated. 2016. ISBN 9781119252245.

[25] OWASP: Top Ten 2017 Project. The Open Web Application Security Project. 2019. [Online; accessed on 19.02.2019].
Retrieved from: `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project`

[26] PAAR, C.; PELZL, J.: Hash Functions. In *Understanding Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg. second edition. 2010. ISBN 9783642041006. pp. 293–317. doi:10.1007/978-3-642-04101-3_11.
Retrieved from: `https://doi.org/10.1007/978-3-642-04101-3_11`

[27] PRENEEL, B.: HMAC. In *Encyclopedia of Cryptography and Security*, edited by H. C. A. van Tilborg. Boston, MA: Springer US. 2005. ISBN 978-0-387-23483-0. pp. 267–268. doi:10.1007/0-387-23483-7_187. [Online, accessed on 15.04.2019].
Retrieved from: `https://doi.org/10.1007/0-387-23483-7_187`

[28] PUDER, A.; RÖMER, K.; PILHOFER, F.: *Distributed Systems Architecture*. San Francisco: Morgan Kaufmann. 2006. ISBN 978-1-55860-648-7. 7 - 31 pp.. doi:https://doi.org/10.1016/B978-1-55860-648-7.X5000-1. [Online, accessed on 18.04.2019].
Retrieved from: `https://doi.org/10.1016/B978-1-55860-648-7.X5000-1`

[29] RESCORLA, E.: The Transport Layer Security (TLS) Protocol Version 1.3. Internet Requests for Comments. August 2018. doi:10.17487/RFC8446. [Online, accessed on 15.04.2019].
Retrieved from: `https://doi.org/10.17487/RFC8446`

[30] STALLINGS, W.: *Cryptography and Network Security: Principles and Practice, Global Edition*. Harlow: Pearson Education Limited. 2016. ISBN 9781292158587.

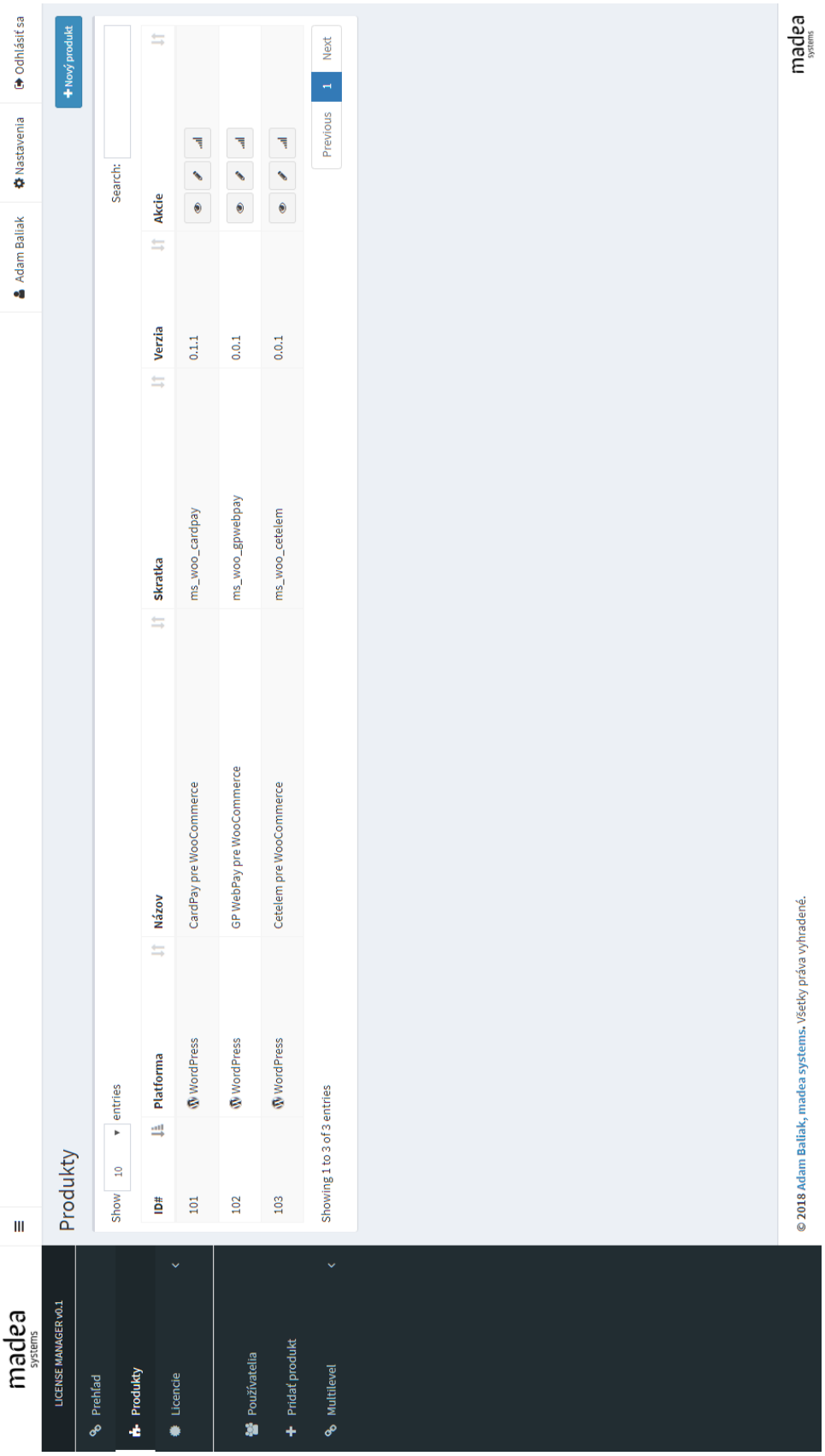
[31] YUSOF, I.; PATHAN, A.: Mitigating Cross-Site Scripting Attacks with a Content Security Policy. *Computer*. vol. 49, no. 3. 2016: pp. 56–63. ISSN 0018-9162. doi:10.1109/MC.2016.76. [Online, accessed on 20.02.2019].
Retrieved from: `https://doi.org/10.1109/SecDev.2016.033`

LIST OF FIGURES

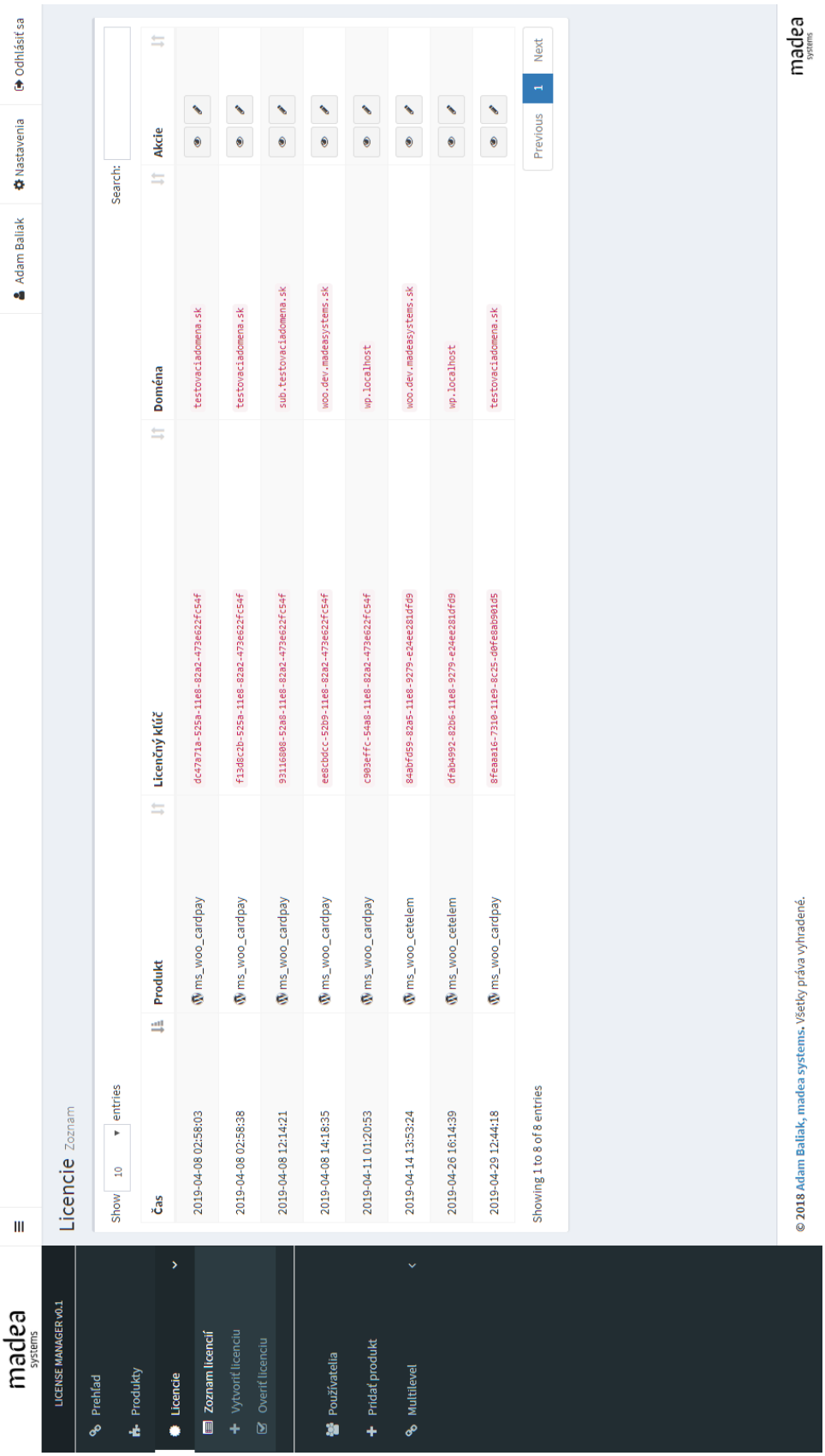
LIST OF TABLES

LIST OF APPENDICES

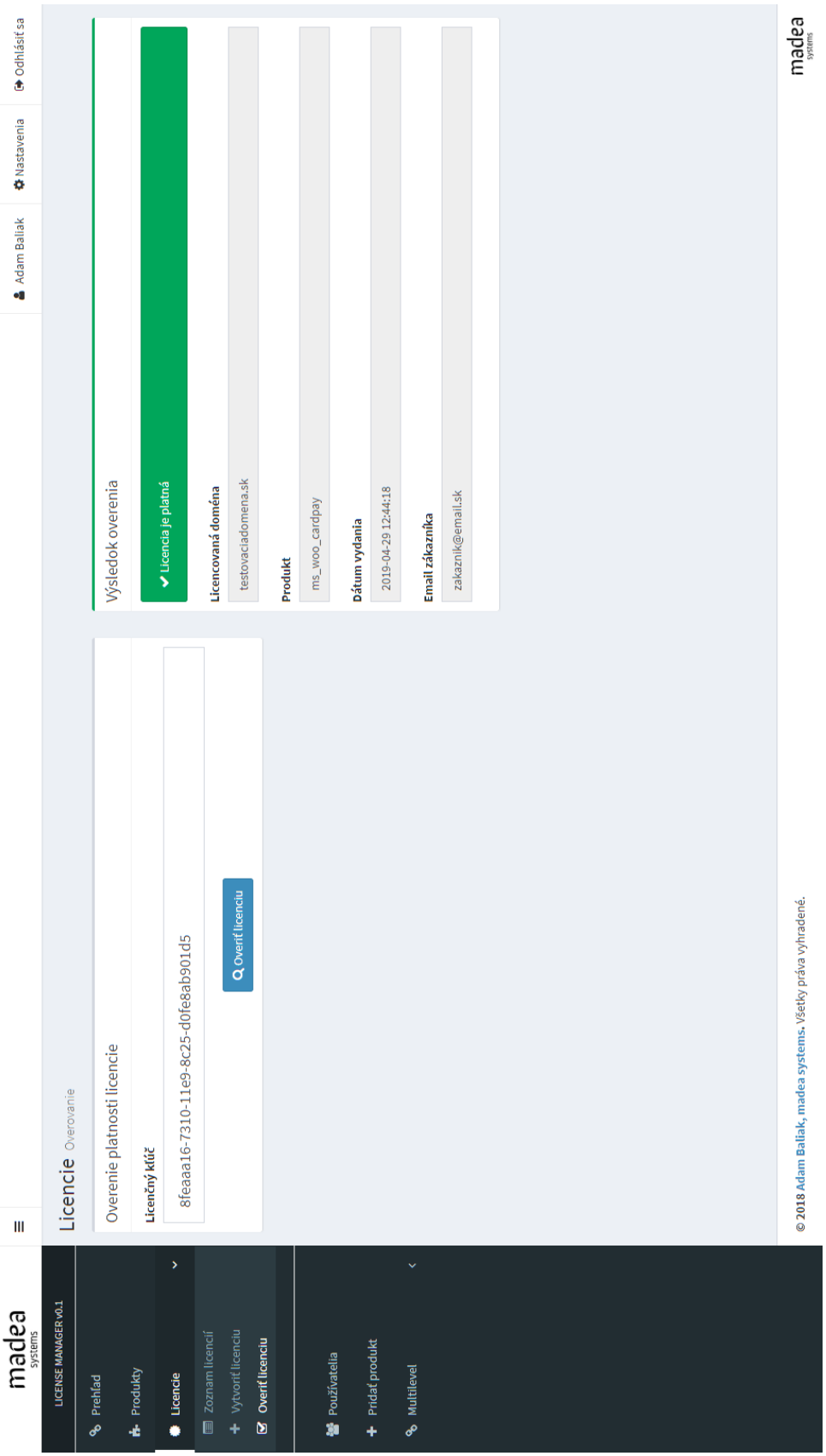
Appendix 1: Product screen



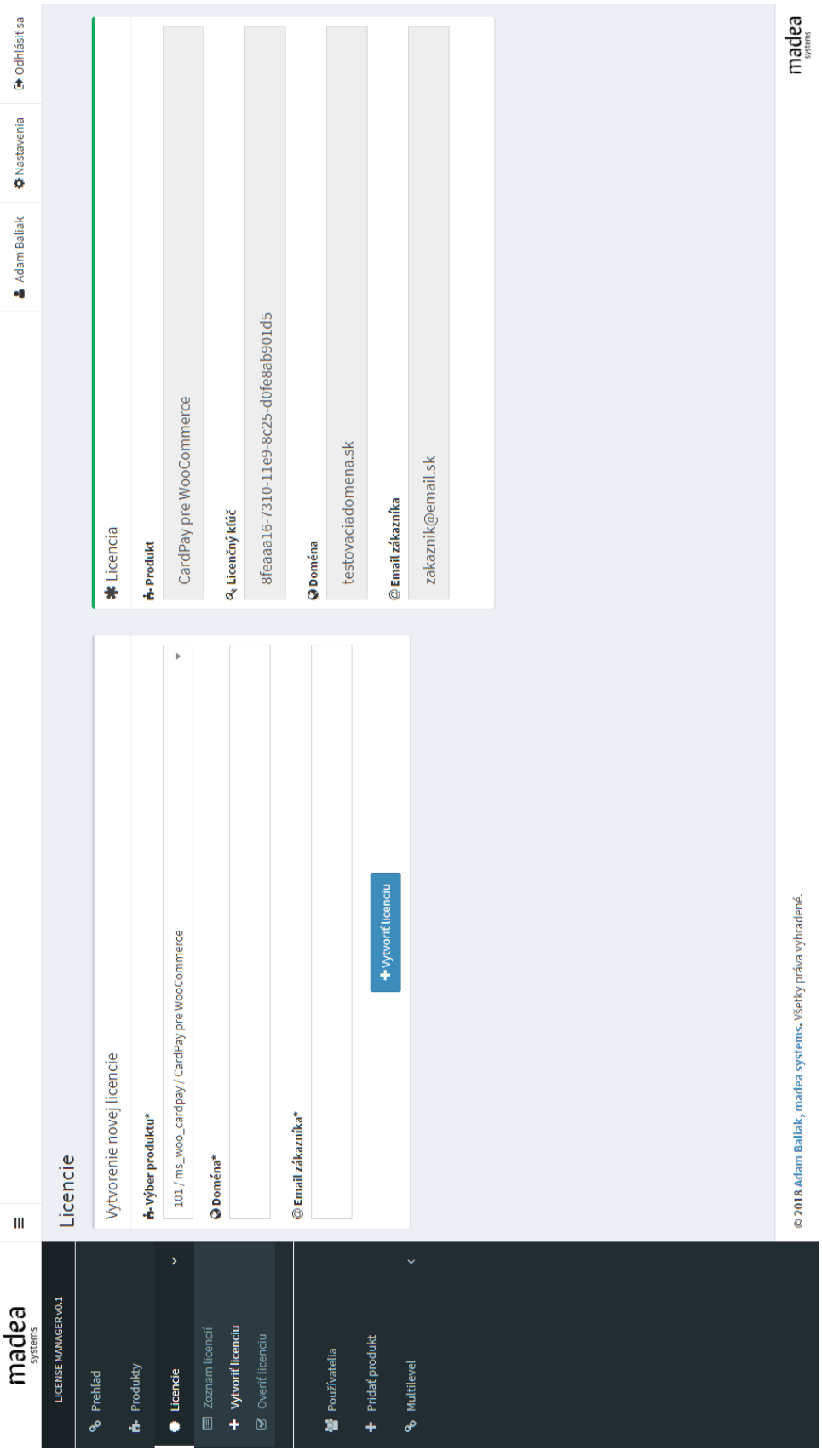
Appendix 2: Licence list screen



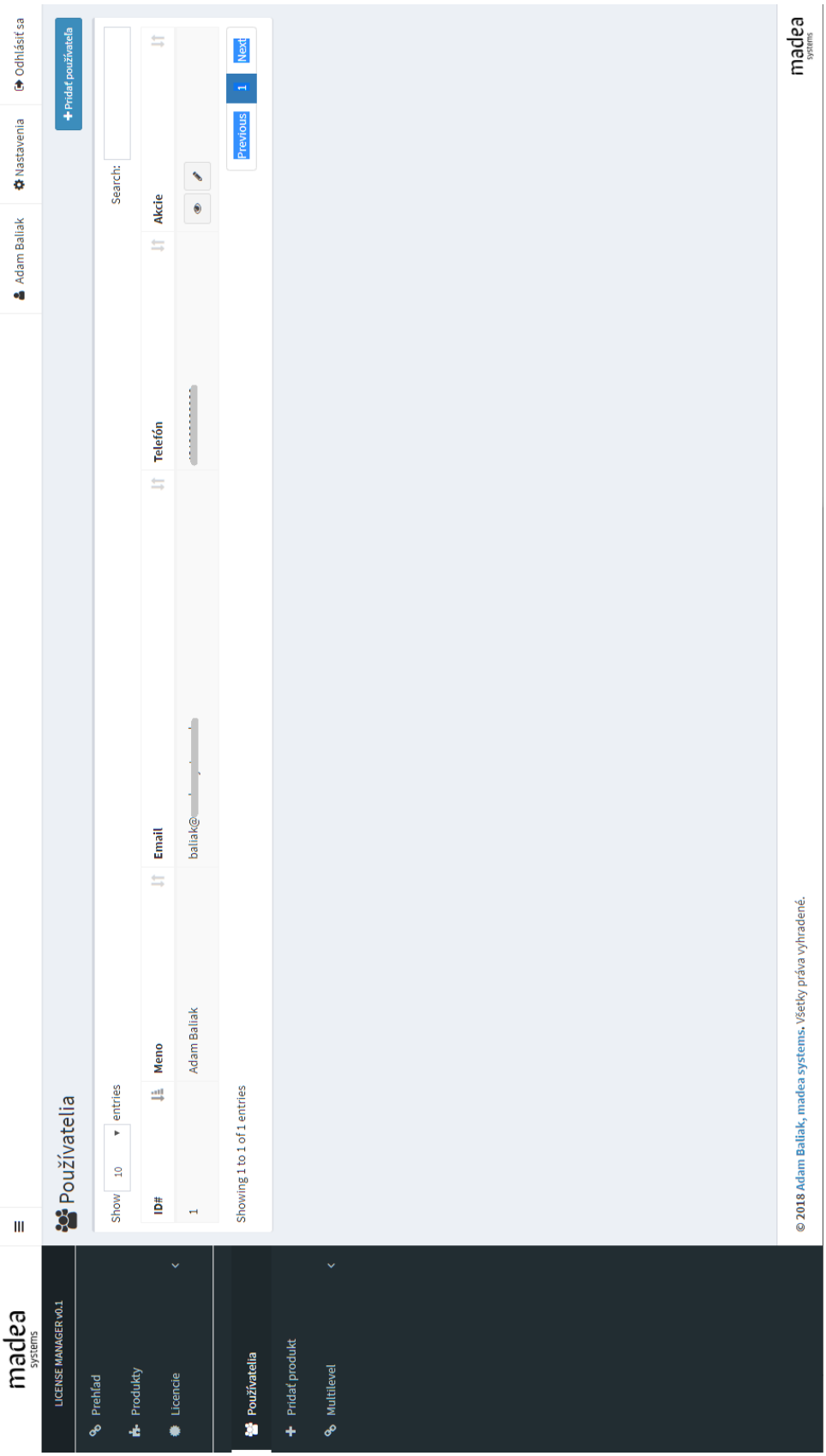
Appendix 3: Verify licence screen



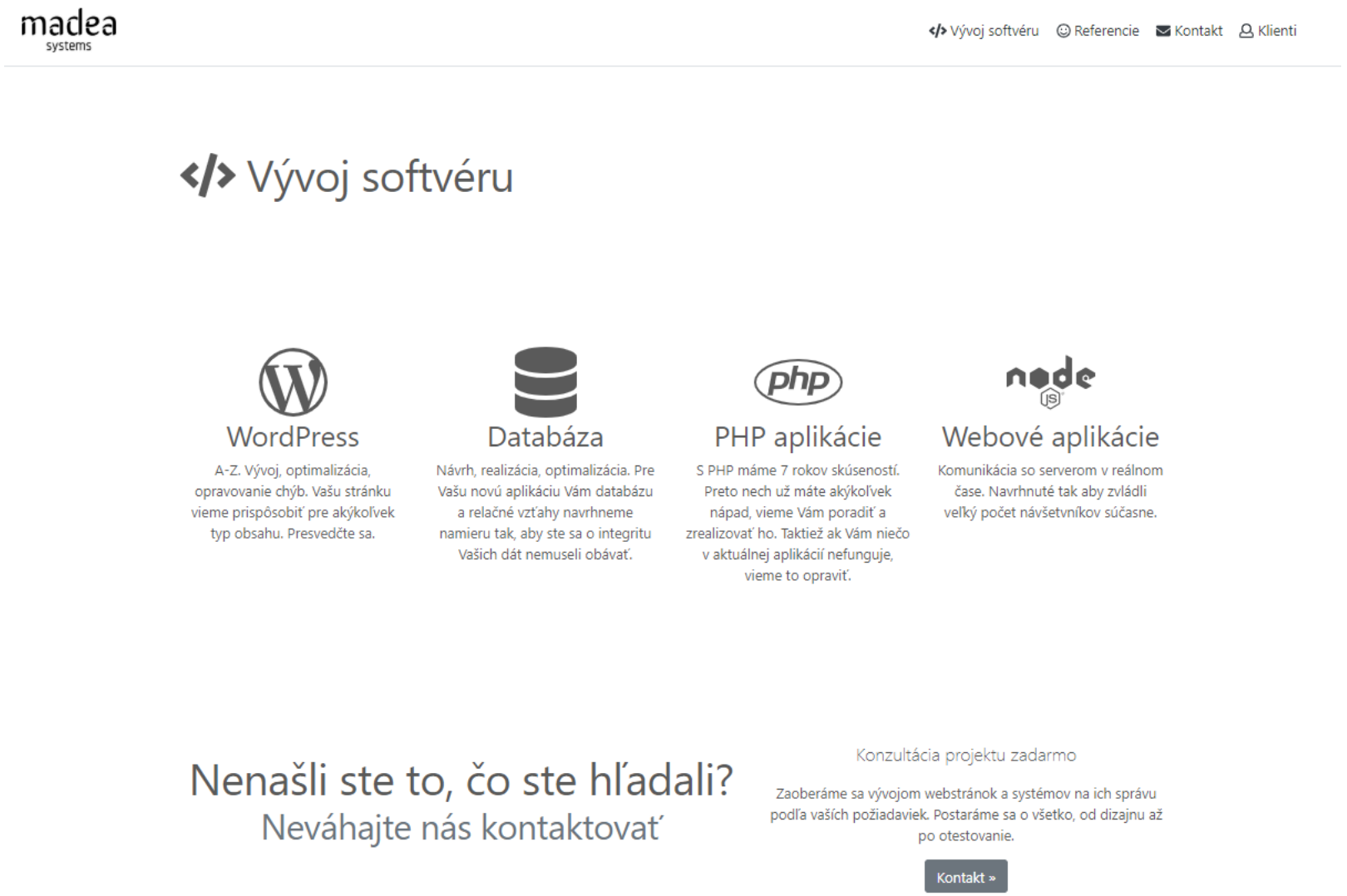
Appendix 4: Issue new licence screen



Appendix 5: User list screen



Appendix 6: User settings screen


madea
systems
Vývoj softvéru
Referencie
Kontakt
Klienti
Vývoj softvéru
WordPress
A-Z. Vývoj, optimalizácia, opravovanie chýb. Vašu stránku vieme prispôsobiť pre akýkoľvek typ obsahu. Presvedčte sa.
Databáza
Návrh, realizácia, optimalizácia. Pre Vašu novú aplikáciu Vám databázu a relačné vzťahy navrhneme namieru tak, aby ste sa o integritu Vašich dát nemuseli obávať.
PHP aplikácie
S PHP máme 7 rokov skúseností. Preto nech už máte akýkoľvek nápad, vieme Vám poradiť a zrealizovať ho. Taktiež ak Vám niečo v aktuálnej aplikácií nefunguje, vieme to opraviť.
Webové aplikácie
Komunikácia so serverom v reálnom čase. Navrhnuté tak aby zvládli veľký počet návšetvníkov súčasne.
Nenašli ste to, čo ste hľadali?
Neváhajte nás kontaktovať
Konzultácia projektu zadarmo
Zaoberáme sa vývojom webstránok a systémov na ich správu podľa vašich požiadaviek. Postaráme sa o všetko, od dizajnu až po otestovanie.
Kontakt »

Appendix 8: Application configuration file

```php
<?php
/**

    MADEA
    SYSTEMS
    SK
        License Management System
        v.0.1.0
        CREATED BY MADEASYSTEMS.SK

*/

//debugging
define('MS_DEBUG', true);

//die if accessed directly
if(!defined('MS_SCRIPT') AND !defined('MS_LCS')){
    header("Location: /");
    die();
}

//set MS enviroment var
if(!defined('MS')){
    define('MS', true);
}

require __DIR__ . '/db.php'; //database
require __DIR__ . '/functions.php'; //functions
require __DIR__ . '/vendor/autoload.php'; //composer requirements

//secrets
define('__FP_SECRET',       'uST%VA1D[{Ys?DMA(%JTf]k@tlVqx$Dm8lSX8422$[_2=s~Ib{7[#0WDTnnC');
define('__COOKIE_SECRET',   '~UbamJ+3mBOCu6DMzyZ@f+yO?18!xqgY~-+[RB0UTcL_Yb!llGF5]cZx8b2J');

//defines
define('DS', DIRECTORY_SEPARATOR);
define('_SERVER_DOMAIN', 'plugins.madeasystems.sk');
define('LCS_PATH', '/lcs/');
define('PHONE_2FA', false);

//auth_keys
define('NEXMO_API_KEY',     '_hidden_');
define('NEXMO_API_SECRET',  '_hidden_');

//private section security
if(defined('LCS_PRIVATE')){

    require __DIR__ . '/cookie.php';
}

if( MS_DEBUG ){

    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL & ~E_NOTICE);
}
```

Appendix 9: Source code of endpoint verifying licence validity

```php
<?php
if( !isset($_POST['x1'])
    OR $_POST['x2'] == ""
    OR !isset($_POST['x1'])
    OR $_POST['x1'] == ""
    OR $_SERVER['HTTP_USER_AGENT'] == "" ){

        die("INVALID_REQUEST");
    }

define('MS_SCRIPT', true);

require_once __DIR__ . '/inc/ms.php';

$q = $db->prepare("SELECT product_uid_by_code(?)");
$q->bind_param("s",$_GET['product_code']);
$q->execute();
$q->bind_result($res);
$q->fetch();

if($res == 'PRODUCT_NOT_FOUND'){

    die('PRODUCT_NOT_FOUND');
} else {

    $product_code = $res;
}
$q->close();

//input variables
$key_hashed  = $_POST['x1']; //sha256 hash of licence_key keyed by fingerprint
$key_client  = $_POST['x2'];
$remote_ip   = $_SERVER['REMOTE_ADDR'];
$remote_host = explode('//', $_SERVER['HTTP_USER_AGENT']);

//calculate fp
$fp_verify = generate_domain_fingerprint(trim($remote_host[1]), $remote_ip, $product_code);
$control_hash = hash_hmac('sha256', $key_client, $fp_verify);

//hash verification
if( !hash_equals($control_hash, $key_hashed)  OR $fp_verify == 'INVALID_REQUEST'){

    //fingerprint is invalid
    die('INVALID_REQUEST');
}

//$remote_host is now trusted
//check license_key validity for that host
$q = $db->prepare("SELECT licence_key_domain_by_key(?)");
$q->bind_param("s",$key_client);
$q->execute();
$q->bind_result($res);
$q->fetch();

if($res == 'KEY_NOT_FOUND'){

    die('KEY_NOT_FOUND');
}

if(trim($res) != trim($remote_host[1])){

    die('INVALID_KEY');
}

die("VALID_LICENCE");

$q->close();
```